实用第一 智慧密集 Practicability First, Intelligence Intensive

电脑编程技巧与维护

COMPUTER PROGRAMMING SKILLS & MAINTENANCE

http://www.comprg.com.cn

上 2月

2012年2月23日

每期定价:11.00元 全年定价264.00元
《电脑编程技巧与维护》杂志社出版
刊号: ISSN 1003-4052 CN11-3411/TP
广告许可证 京海工商广字0151

国家级科技期刊 中国学术期刊综合评价数据库统计源期刊 中国核心期刊(遴选)数据库收录期刊

ISSN 1003-4052

欢迎订阅

数字通信世界

DIGITAL COMMUNICATION WORLD

ISSN 1672-7274
CN 11-5154/TN

《数字通信世界》（刊号：ISSN 1672-7274 CN 11-5154/TN）是我国综合报道通信、卫星、广电、航天领域的科技综合月刊。

《数字通信世界》秉承"为技术创新和企业发展服务"的宗旨，立足数字通信产业，积极宣传报道宏观产业政策，密切关注通信、卫星、广播、电视、互联网等行业新动态；及时传播现代企业管理观念；大胆预测数字通信市场、技术、运营、业务发展趋势；致力探索、推介通信行业创新经营模式、新技术、新设备、新产品；

《数字通信世界》以权威、新颖、实用信息回报广大读者的厚爱。向各级决策与经营管理人员、市场营销及采购人员、科研开发与技术支持人员提供新的国内外技术与产品、市场信息、业内动态及管理经验；为树立制造业与运营业客户的企业形象、提高知名度、传播产品信息、沟通供销渠道提供信息平台。本刊作为专业信息媒体，努力将自己打造成用户的益友、企业的桥梁。集科技、工程、应用、市场、经营、法规于一体，以一流的质量和服务推动行业的发展，成为企业成长的助手和广大读者的朋友。密切关注行业的新动态，始终站在行业的前沿，推动行业持续良性发展是我们义不容辞的责任！

杂志订阅方式：

A: 邮局汇款
收 款 人：肖红
通信地址：北京市173信箱《数字通信世界》杂志社
邮政编码：100036
优惠订阅价：120元/年（含赠品）

B: 银行汇款
开 户 行：招商银行北京分行西三环支行
帐　　号：2082912610001
户　　名：北京数通广告有限责任公司
优惠订阅价：120元/年（含赠品）

C: 邮局订阅
邮发代号：80-393
订 阅 价：120元/年（不含赠品）

服务产业发展的信息平台
探索解决方案的专家论坛

请订阅

订　阅	单 价	起止期数	份 数	金 额
杂　志	10元/期	年　期至　期		
合 订 本	150元/本	年		
卫星资源图	10元/张			
卫星电视节目图	10元/张			
姓　名			手 机	
单位名称			电 话	
邮寄地址			邮 编	
Email			传 真	

请订户将此表填好后传真或寄回本刊（附带汇款单复印件）

与本刊发行部直接订阅《数字通信世界》杂志

赠

《数字通信世界》期刊电子档
亚太地区卫星资源指南图
亚太地区卫星电视频道指南图

邮寄地址：北京市万寿路173信箱《数字通信世界》杂志社（100036）
电　　话：+86-10-88254338/4339/4347/4348
传　　真：+86-10-88254349
网　　址：www.dcw.org.cn　www.dcw.net.cn

2013 年第 03 期
2 月(上)

电脑编程技巧与维护

总第 273 期　　1994 年 7 月创刊　　(半月刊)

编 辑 出 版：《电脑编程技巧与维护》杂志社
主 管 部 门：中华人民共和国工业和信息化部
主 办 单 位：中国信息产业商会
地　　址：北京市海淀区长春桥路 5 号
6 号楼 1209 室
投 稿 信 箱：gaojian@comprg.com.cn
gaojian@comprg.sina.net
编辑部信箱：gaojian@comprg.com.cn
发行部信箱：zzsfx@vip.sina.com
网　　址：http://www.comprg.com.cn
邮　　编：100089
电　　话：(010) 82561037
传　　真：(010) 82561614
照　　排：《电脑编程技巧与维护》
杂志社电脑排版部
印 刷 厂：北京慧美印刷有限公司
订 阅 处：全国各地发行局
国内总发行：北京报刊发行局
邮 发 代 号：82-715
国外发行代号：M6232
刊　　号：ISSN 1006-4052 / CN11-3411/TP
广告订可证：京海工商广字 0151 号
全 年 定 价：264 元
每 期 定 价：11 元

电脑编程技巧与维护(半月刊)

每月3日出版

实用第一 智慧密集

COMPUTER PROGRAMMING SKILLS & MAINTENANCE

目次

Atheros 推出新技术为 802.11ac 全面提速

高通 Atheros 宣布推出创新的 StreamBoost 技术，它将帮助采用高通 802.11ac 芯片的无线路由器合理分配每一款接入终端的带宽，为用户带来更加出色的 11ac 千兆无线应用体验。

随着智能移动终端的爆发式增长，目前现代家庭中至少都有 4、5 台无线终端需要接入到家庭的无线网络之中，而且这一数字在未来几年中还会进一步提升。虽然 11ac 的出现，使得无线网络的速率提升至 1300Mbps，但相信每一位无线用户都有过这样的体验——当家中的这些智能终端同时接入到无线网络时，一旦有某个终端开启了高带宽应用，其他终端的网络应用体验就会变得很差，而 StreamBoost 技术的出现，将轻松解决用户的这一烦恼。

通过 StreamBoost 技术，每一款接入无线网络的终端设备都将会被自动分配合理的带宽，例如，一个终端正在进行在线游戏，而另一个终端则只是进行简单的网页浏览，那么 StreamBoost 就会智能的为在线游戏的终端分配更多的带宽，以保证其应用的流畅性，同时也会保证浏览网络必须的足够带宽。

通过 StreamBoost 的客户端，用户还可以清晰的看到目前所有接入网络的设备的类型，以及它们所需的带宽和正在使用的带宽。而且它还提供了“云”服务，它将通过识别终端类型，自动为接入终端分配最新、最合理的带宽。

最快移动处理器 Tegra 4 亮相 支持 4G LTE 语音与数据

在 2013 CES 上，NVIDIA® (英伟达 ™) 公司发布全球最快的移动处理器 NVIDIA® Tegra® 4，其创纪录的性能与电池续航时间完美支持智能手机、平板电脑、游戏设备、汽车信息娱乐系统、导航系统以及 PC。

Tegra 4 具备超强的图像处理能力，可实现闪电般的 Web 浏览速度、具有震撼力的图形效果以及全新的计算照相功能。

原代号为“Wayne”的 Tegra 4 具备 72 个定制 NVIDIA® GeForce® GPU 核心，其 GPU 处理性能是 Tegra 3 的六倍，可实现更加逼真的游戏体验和更高分辨率的屏幕显示效果。Tegra 4 首次集成应用了四颗 ARM 最先进 Cortex-A15 CPU 核心，Web 浏览可实现 2.6 倍的速度提升，应用程序的性能也可实现飞跃。

Tegra 4 还可通过选装芯片组——第五代 NVIDIA Icera® i500 处理器实现全球 4G LTE 语音与数据致支持。Icera® i500 处理器效率更高，处理速度是传统调制解调器的四倍，但体积却只有其 40%。

NVIDIA 移动事业部的高级副总裁 Phil Carmack 表示：“Tegra 4 具备超强的处理能力和超高的效率，为智能手机、平板电脑、游戏设备、汽车系统以及 PC 提供了强劲的动力支持。其全新功能，尤其是在计算照相方面的创新能够极大地提升众多现有产品的性能，并可催生新的产品。”

计算照相能力

Tegra 4 众多的突破性创新之中包含摄影运算架构，它通过融合 GPU、CPU 与相机图像信号处理器的处理能力，可自动支持高动态范围 (HDR) 照片与视频。

HDR 技术可完美捕捉图像，其中包括使用闪光灯拍摄的图像，与人肉眼看到的效果完全一致，明亮与昏暗区域的细节表现十分完美。

前所未有的电能效率

Tegra 4 具备一个第二代节电核心，具备极高的能源效率，适合低功耗的标准使用环境，其 PRISM 2 Display 技术可以在实现高级视觉效果的同时减少背光功耗。

在一般使用环境中，Tegra 4 的消耗功率比前一代的 Tegra 3 减少了 45%。同时，它可实现最长 14 小时的手机高清视频播放。

Tegra 4 主要特性

- 具备 72 个 GeForce GPU 核心
- 四核 ARM Cortex-A15 CPU 以及一个第二代节电核心
- 摄影运算架构
- 通过可选的 Icera i500 芯片组可支持 LTE 功能
- 4K 超高分辨率视频支持

谷歌将研发接近人类思维搜索引擎：代替人想法

世界知名人工智能专家、谷歌工程部新任总监雷·库兹韦尔 (Ray Kurzweil) 透露，他希望打造一款极为先进的搜索引擎，它就像一个接近人类思维的机器人，比用户自己更了解他们。

库兹韦尔表示：“我设想，若干年后，你实际上还没进行搜索查询就已经得到问题的答案了。”

库兹韦尔说：“这本质上是语言问题。”他认为，真正的人工大脑能够理解思想和概念的意思，语言是创造这种大脑的窗口。“你写一篇博客，不是简单的词语堆积，而是创造一些有意义的语句。”目前，搜索引擎是依靠暴力算法从热门页面中挑选出关键词，希望结果能生成最佳信息。

所谓的“语义”搜索能解析词语背后的含义和意图，意在解决“热狗”问题。正如谷歌董事长埃里克·施密特的解释：“它是‘hot dog’ (一条很热的狗) 还是‘hotdog’ (热狗面包)？如果知道某个人是否养狗或者是否为素质主义者，就可能会对该问题做出截然不同的回答。”

最终，谷歌将了解用户为何搜索信息并为他们提供自己都不知道需要的答案。这种全能的新思维教育需要谷歌庞大的数据库作为支撑。库兹韦尔说，谷歌比任何公司都更了解“你在邮件、博客、乃至聊天中的所读、所写、所闻、所说。”

谷歌可以综合搜索引擎的个性化建议，创造一个无比了解用户的超级好友。它知道你的健康问题、商业战略。此外，它能仔细审阅、筛选世界各地每一分钟产生的最新信息，然后自动推送给你。

基于粒子群算法的迷宫路径搜索问题研究

刘 烽

摘 要：粒子群算法是一种很好的优化工具，提出了针对迷宫问题求解的粒子群编码和种群进化规则。通过对一个具体实例的求解，说明粒子群算法对于求解迷宫问题具良好的效果。

关键词：粒子群算法；迷宫问题；路径搜索

1 引言

迷宫问题如图 1 所示是实验心理学中一个古典问题，以 m*n 的长方阵表示迷宫，0 和 1 分别表示迷宫中的通路和障碍，给定迷宫的入口和出口。要求求出迷宫入口到出口有无通路，若有通路则指出其中一条通路和路径。迷宫问题是典型的图搜索问题，解决方法通常有启发式搜索和 A* 算法，鉴于粒子群算法 (Particle Swarm Optimization,PSO) 良好的优化能力，下文尝试利用粒子群算法求解迷宫问题。

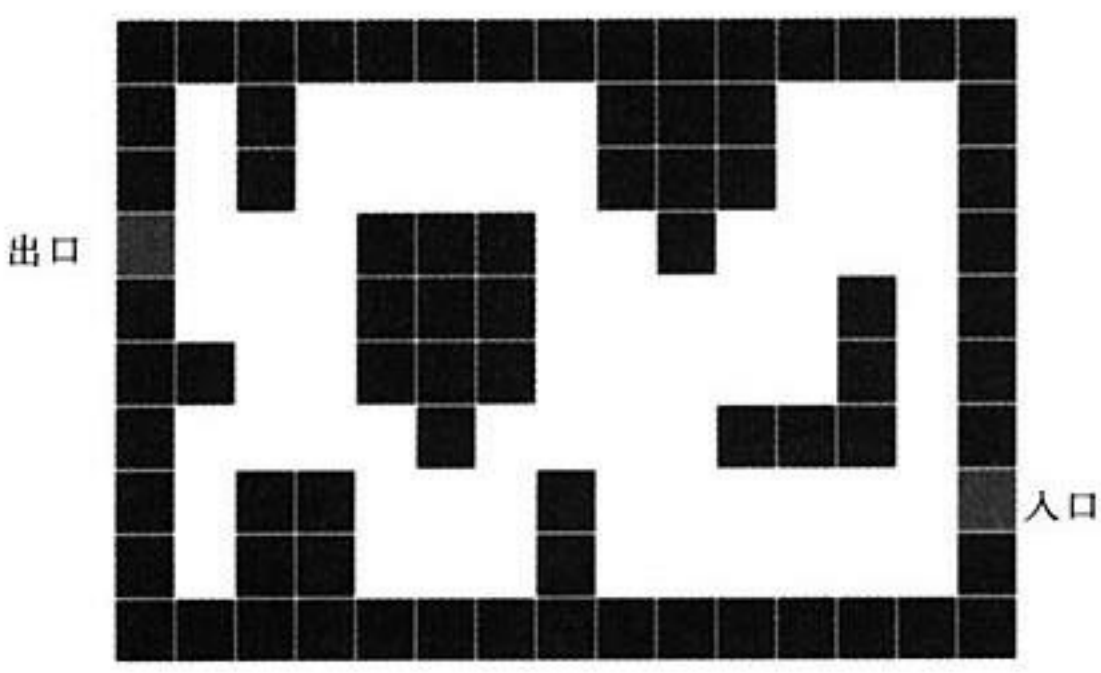


图 1 迷宫问题

在 PSO 中，每个优化问题的解都是搜索空间中的一只鸟，称之为“粒子”。所有粒子都有一个由被优化的函数决定的适应值，每个粒子还有一个速度决定它们飞翔的方向和距离，粒子们就追随当前的最优粒子在解空间中搜索。PSO 初始化为一群随机粒子（随机解）。然后通过迭代找到最优解。每次迭代过程通过跟踪两个“极值”来更新自己。第一个就是粒子本身所找到的最优解。这个解叫做个体极值 pBest，另一个极值是整个种群目前找到的最优解。这个极值是全局极值 gBest。粒子根据式 (1) 和式 (2) 来更新自己速度和新的位置：

$$V_{id} = w \times V_{id} + c_1 \times rand( ) \times (p_{id} - x_{id}) + c_2 \times rand( ) \times (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + V_{id} \quad (2)$$

其中，V_{id} 是粒子的速度，P_{id} 是当前粒子的位置。c_1 和 c_2 为学习因子，w 为惯性系数。

2 设计思路

2.1 粒子编码

PSO 算法一般采用实数编码，对于图 1 的迷宫，从入口到出口的路径共有 4 个方向可选择：向上、向下、向左、向右。那么用 1，2，3，4 分别代表，上、下、左、右 4 个方向，假设有随机数 P，若 $P \in [1,2]$，则选择向上方向运动，依次类推。那么，粒子的位置 x_{id} 代表入口向出口的一条路径（不一定是通路），例如有粒子 $particle = [3.1,3.5,2.4,3.2,\cdots]$，这条路径为向左→向左→向下→向左等。

2.2 适应值的计算

适应值的计算可参考曼哈顿算法，即当前位置距出口距离的倒数，具体计算方法为 $FitScore = \frac{1}{Dis}$，其中 $Dis = abs(DestinationX - CurrentX) + abs(DestinationY - Currenty) + 1$（距离的基数为 1），由于到达终点时，距出口的距离为 0，适应度值为最大值 1。因此，在整个搜索过程中，粒子的搜索目标始终朝着适应度为 1 的方向搜索。

2.3 种群初始化

粒子的初速度设为 0，每个个体局部极值 pBest 的初值设为每个粒子自身，粒子的全局极值 gBest 初始值为第一代粒子群的最优值。

2.4 粒子更新及参数选择

粒子速度和位置的更新基本按式 (1) 和式 (2)，为保证粒子位置和速度的合法性，将式 (2) 取绝对值 [1,4]，同时由于粒子的速度取值在内，假定：当粒子速度小于 1 或大于 4 时，那么粒子速度就设定为 1 或 4。需要说明的是，需要预先估计从入口到出口的最大步数，对于图 2 的迷宫，假设从入口到出口最多需要 35 步，在这里对应粒子的长度。

3 实现代码

算法实现采取 Win32 编程，程序在 VC 6.0+WindowsXP 环境中调试通过，程序运行结果如所图 2 示。

图 2 PSO 求解迷宫问题结果图

算法实现过程中的核心类主要有：粒子类、算法类和绘图类。

```
//////////////////////////////////////////////////////////////////////
//
//          文件: CParticle.h
//          描述: 粒子类
//
//////////////////////////////////////////////////////////////////////
#ifndef CPARTICLE_H
#define CPARTICLE_H
#include <vector>
#include <sstream>
#include "defines.h"
#include "CMap.h"
#include "utils.h"
using namespace std;
class CParticle
{
public:
    //粒子位置
    vector<float> vecBits;
    //粒子速度
    vector<float> vecVelocity;
    //适应值
    double dFitness;
    CParticle():dFitness(0)
    {}
    CParticle(const int num_bits):dFitness(0)
    {
        for (int i=0;i<=num_bits;i++)
        {
            //生成(1,5)之间的随机数
            vecBits.push_back(RandFloat(1,5));
            //粒子初速度为 0
            vecVelocity.push_back(0.0);
        }
    }
};
#endif
//////////////////////////////////////////////////////////////////////
//
//          文件: CPSO.h
//          描述: 粒子群算法类
//
//////////////////////////////////////////////////////////////////////
#ifndef CPSO_H
#define CPSO_H
#include "CPSO.h"
#include "utils.h"
#include "CParticle.h"
class CPSO
{
private:
    //种群
    vector<CParticle>    m_vecParticles;
    //局部极值的粒子
    vector<CParticle>    m_vecLocBest;
    //种群大小
    int  m_iPopSize;
    //PSO 粒子更新参数
    float m_C1;
    float m_C2;
    float m_w;
    //粒子编码长度
    int  m_iParticleLength;
    //全局最优的个体
    int m_iGlobalBest;
    double m_dBestFitnessScore;
    //进化的当前代数
    int m_iGeneration;
    //迷宫
    CMap  m_Map;
    //通路
    CMap m_Path;
    bool  m_bBusy;
    void  CreateStartPopulation();
    void  UpdateFitnessScores();
    vector<int>FloatToInt(const vector<float> &vec);
protected:
public:
    CPSO(int popsize,
         double c1,
         double c2,
```

FOLLOW MASTER PROGRAM

```
                double w,
                int num_bits):m_iPopSize(popsize),
                                m_C1(c1),
                                m_C2(c2),
                                m_w(w),
                                m_iGeneration(0),
                                m_dBestFitnessScore(0.0),
                                m_iParticleLength(num_bits),
                                    m_bBusy(false)
        {
                CreateStartPopulation();
        }

void        Run(HWND hwnd);
void        Render(int cxClient, int cyClient, HDC surface);
        void  Epoch();
        int  Generation(){return m_iGeneration;}
        int  GetFittest(){return m_iGlobalBest;}
        bool  Started(){return m_bBusy;}
        void  Stop(){m_bBusy = false;}
};
#endif
//////////////////////////////////////////////////////////////////////
//
//          文件: CPSO.cpp
//
//          描述: 粒子群算法类实现
//
//////////////////////////////////////////////////////////////////////
#include "CPSO.h"
//------------------------------
CreateStartPopulation ------------
//    粒子初始化,产生初始种群
//------------------------------------------
void CPSO::CreateStartPopulation()
{
        m_vecParticles.clear();

        for (int i=0; i<m_iPopSize; i++)
        {
                        m_vecParticles.push_back (CParticle
(m_iParticleLength));
        }
        m_vecLocBest = m_vecParticles;
        m_iGeneration = 0;
        m_iGlobalBest = 0;
        m_dBestFitnessScore = 0;
}
//------------------- Epoch -----------------
//    粒子根据算法规则进行进化
//-----------------------------------------
void CPSO::Epoch()
{
        double temp;
        CMap TempMemory;
        UpdateFitnessScores();
        for (int i=0; i < m_iPopSize; ++i)
        {
                for (int j = 0; j < m_iParticleLength; ++j)
                {
                        m_vecParticles  [i].vecVelocity  [j] = m_w *
m_vecParticles[i].vecVelocity[j] +
                                m_C1 * RandFloat() * ( m_vecLocBest
[i].vecBits[j] – m_vecParticles[i].vecBits[j] ) +
                                m_C2 * RandFloat() * ( m_vecParticles
[m_iGlobalBest].vecBits[j] – m_vecParticles[i].vecBits[j]);
                        temp = abs ( m_vecParticles  [i].vecBits  [j] +
m_vecParticles[i].vecVelocity[j]);
                        if ( temp <= 1.0)
                        {  .
                                m_vecParticles[i].vecBits[j] = 1.0;
                        }
                        else if ( temp >= 4.0)
                        {
                                m_vecParticles[i].vecBits[j] = 4.0;
                        }
                        else
                        {
                                m_vecParticles[i].vecBits[j] = temp;
                        }
                }
                //update local best
                        vector<int> vecDirections = FloatToInt
(m_vecParticles[i].vecBits);
                m_vecParticles  [i].dFitness  =  m_Map.TestRoute
(vecDirections,TempMemory);
                if  (m_vecParticles  [i].dFitness > m_vecLocBest[i].
dFitness)
                {
                        m_vecLocBest[i] = m_vecParticles[i];
                }
        }

        ++m_iGeneration;

}
//--------------- UpdateFitnessScores ------------
//    更新局部极值和全局极值
//-----------------------------------------
void CPSO::UpdateFitnessScores()
{
        m_dBestFitnessScore = 0;
        m_dBestFitnessScore = 0;
        m_iGlobalBest = 0;
        CMap TempMemory;
```

```
    for (int i=0; i < m_iPopSize; ++i)
    {
                    vector<int> vecDirections = FloatToInt
(m_vecParticles[i].vecBits);
        m_vecParticles [i].dFitness = m_Map.TestRoute
(vecDirections,TempMemory);
                    if (m_vecParticles [i].dFitness >
m_dBestFitnessScore)
        {
    m_dBestFitnessScore = m_vecParticles[i].dFitness;
            m_iGlobalBest = i;
            m_Path = TempMemory;
            if (m_vecParticles[i].dFitness == 1)
            {

                m_bBusy = false;
            }
        }

        TempMemory.ResetMemory();

    }
}
//----------------- FloatToInt -----------------
//    根据粒子种群生成路径编码
//----------------------------------------
vector<int> CPSO::FloatToInt(const vector<float> &vec)
{
    vector<int> vecDirection;
    for (int i = 0 ; i < vec.size() ; i++)
    {
        vecDirection.push_back(floor(vec[i]));
    }
    return vecDirection;
}
void CPSO::Run(HWND hwnd)
{
    CreateStartPopulation();
    m_bBusy = true;
}
//---------------- Render ----------------
//    打印算法运行的参数信息及算法的运行及停止
//----------------------------------------
void CPSO::Render(int cxClient, int cyClient, HDC surface)
{
    m_Map.Render(cxClient, cyClient, surface);
    m_Path.MemoryRender(cxClient, cyClient, surface);
    string s = " 当前代数: " + itos (m_iGeneration) + " :PSO
算法参数:W=" + doubletos(W) + ",C1=" + doubletos(C1) + ",
C2=" + doubletos(C2) + " 种群大小:" + itos(POP_SIZE);
    TextOut(surface, 5, 0, s.c_str(), s.size());
    if (! m_bBusy)
    {
        string Start = "回车键开始...";
        TextOut (surface, cxClient/2 - (Start.size () * 3),
cyClient - 20, Start.c_str(), Start.size());
    }
    else
    {
        string Start = "空格键停止";
        TextOut (surface, cxClient/2 - (Start.size () * 3),
cyClient - 20, Start.c_str(), Start.size());
    }
}
//////////////////////////////////////////////////////////
//
//        文件: CMap.h
//        描述: 定义迷宫类
//
//////////////////////////////////////////////////////////
#ifndef CMAP_H
#define CMAP_H

#include "stdlib.h"
#include "windows.h"
#include <vector>
#include "defines.h"
using namespace std;
class CMap
{
private:
    //迷宫
    static const int  map[MAP_HEIGHT][MAP_WIDTH];
    static const int  m_iMapWidth;
    static const int  m_iMapHeight;
    //迷宫起点
    static const int  m_iStartX;
    static const int  m_iStartY;
    //迷宫终点
    static const int  m_iEndX;
    static const int  m_iEndY;
public:
    int    memory[MAP_HEIGHT][MAP_WIDTH];
    CMap()
    {
        ResetMemory();

    }
    //测试路径,同时计算适应度值
    double TestRoute (const vector<int> &vecPath, CMap
&memory);
    //绘制迷宫
    void Render (const int cxClient, const int cyClient, HDC
surface);
```

FOLLOW MASTER PROGRAM

```
    //绘制通路
    void MemoryRender (const int cxClient, const int
cyClient, HDC surface);
    //清空路径
    void ResetMemory();
};

#endif
//////////////////////////////////////////////////////////////////
//
//          文件: CMap.h
//          描述: 迷宫类实现
//
//////////////////////////////////////////////////////////////////
#include "CMap.h"

//用矩阵表示迷宫,1 表示可经过点,0 表示不可经过点
const int CMap::map[MAP_HEIGHT][MAP_WIDTH] =
{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
 8, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1,
 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1,
 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1,
 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 0, 1,
 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1,
 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,
 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1,
 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1,
 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 5,
 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1};

const int CMap::m_iMapHeight = MAP_HEIGHT;
const int CMap::m_iMapWidth  = MAP_WIDTH;
const int CMap::m_iStartX = 14;
const int CMap::m_iStartY = 16;
const int CMap::m_iEndX = 0;
const int CMap::m_iEndY = 3;
//---------------- Render ----------------
//    绘制迷宫
//----------------------------------------
void CMap::Render(const int cxClient,
                  const int cyClient,
                  HDC surface)
{
    const int border = 20;
    int BlockSizeX = (cxClient - 2*border)/m_iMapWidth;
    int BlockSizeY = (cyClient - 2*border)/m_iMapHeight;
    HBRUSH  RedBrush, OldBrush;
    HPEN    NullPen, OldPen;
    HBRUSH  hBlueBrush;
    hBlueBrush = CreateSolidBrush(RGB(0,0,255));
    NullPen = (HPEN)GetStockObject(NULL_PEN);
    RedBrush = CreateSolidBrush(RGB(255,0,0));
OldBrush = (HBRUSH)SelectObject(surface, hBlueBrush);
    OldPen  =  (HPEN)SelectObject(surface, NullPen);
for (int y=0; y<m_iMapHeight; ++y)
{
    for (int x=0; x<m_iMapWidth; ++x)
    {
        int left  = border + (BlockSizeX * x);
        int right = left + BlockSizeX;
        int top   = border + (BlockSizeY * y);
        int bottom = top + BlockSizeY;
        if (map[y][x] == 1)
        {
            SelectObject(surface, hBlueBrush);
    Rectangle(surface, left, top, right, bottom);
        }
        if ( (map[y][x] == 5) || (map[y][x] == 8))
        {
            SelectObject(surface, RedBrush);
            Rectangle(surface, left, top, right, bottom);
            }
        }
    }
    SelectObject(surface, OldBrush);
    SelectObject(surface, OldPen);
    DeleteObject(hBlueBrush);
}
//---------------- MemoryRender ----------------
//    绘制入口到出口的通路
//----------------------------------------------
void CMap::MemoryRender(const int cxClient,
                        const int cyClient,
                        HDC surface)
{
    const int border = 20;
    int BlockSizeX = (cxClient - 2*border)/m_iMapWidth;
    int BlockSizeY = (cyClient - 2*border)/m_iMapHeight;
    HBRUSH  OldBrush;
    HPEN    NullPen, OldPen;
    //通路用绿色小方块绘制
HBRUSH hGreenBrush = CreateSolidBrush(RGB(0,255,0));
    NullPen = (HPEN)GetStockObject(NULL_PEN);
OldBrush = (HBRUSH)SelectObject(surface, hGreenBrush);
    OldPen=(HPEN)SelectObject(surface, NullPen);
    for (int y=0; y<m_iMapHeight; ++y)
    {
```

```
for (int x=0; x<m_iMapWidth; ++x)
{

        int left  = border + (BlockSizeX * x);
        int right = left + BlockSizeX;
        int top    = border + (BlockSizeY * y);
        int bottom = top + BlockSizeY;
        if (memory[y][x] == 1)
        {
            Rectangle(surface, left, top, right, bottom);
                }
            }
      }
      SelectObject(surface, OldBrush);
      SelectObject(surface, OldPen);
      DeleteObject(hGreenBrush);
}
//---------------- TestRoute -----------------
//    适应度值的计算
//--------------------------------------------
double CMap::TestRoute(const vector<int> &vecPath, CMap
&Bot)
{
      int posX = m_iStartX;
      int posY = m_iStartY;

for (int dir=0; dir<vecPath.size(); ++dir)
{
      int NextDir = vecPath[dir];
      switch(vecPath[dir])
      {
      case 1: //向上
if ( ((posY-1) < 0 ) || (map[posY-1][posX] == 1) )
            {
                  break;
            }

            else
            {
                  posY -= 1;
            }
            break;
      case 2: //向下
            if   (   ((posY+1) >= m_iMapHeight) || (map
[posY+1][posX] == 1) )
            {
                  break;
            }

            else
            {
                  posY += 1;
            }
            break;
      case 3: //向右
            if   (   ((posX+1) >= m_iMapWidth ) || (map
[posY][posX+1] == 1) )
            {
                  break;
            }

            else
            {
                  posX += 1;
            }
            break;
      case 4: //向左
if ( ((posX-1) < 0 ) || (map[posY][posX-1] == 1) )
            {
                  break;
            }

            else
            {
                  posX -= 1;
            }
            break;

            }
            Bot.memory[posY][posX] = 1;
      }
      int   DiffX = abs(posX - m_iEndX);
      int DiffY = abs(posY - m_iEndY);
      return 1/(double)(DiffX+DiffY+1);
}
//---------------- ResetMemory ----------------
//    复位
//---------------------------------------------
void CMap::ResetMemory()
{
      for (int y=0; y<m_iMapHeight; ++y)
      {
            for (int x=0; x<m_iMapWidth; ++x)
            {
                  memory[y][x] = 0;
            }
      }
}
////////////////////////////////////////////////////////////////
//
//          文件: main.cpp
//          描述: 主函数
//
////////////////////////////////////////////////////////////////
```

FOLLOW MASTER PROGRAM

```
#include <windows.h>
#include <stdlib.h>
#include <time.h>
#include "CPSO.h"
#include "defines.h"
using namespace std;
//////////////////////全局变量 ////////////////////////////////
char*  szApplicationName = "粒子群算法求解迷宫问题";
char*  szWindowClassName = "PathFind";
CPSO* g_pPSO;
//////////////////////////////////////////////////////////////
LRESULT CALLBACK WindowProc(HWND hwnd,
                            UINT msg,
                    WPARAM wparam,
                    LPARAM lparam)
{

    HDC hdc;
    PAINTSTRUCT    ps;
    static int cxClient, cyClient;
    static HDC  hdcBackBuffer;
    static HBITMAP  hBitmap;
    static HBITMAP  hOldBitmap;
    switch(msg)
    {
  case WM_CREATE:
        {
            srand((unsigned) time(NULL));
            g_pPSO = new CPSO (POP_SIZE,C1,C2,W,
PARTICLE_LENGTH);
            RECT rect;
            GetClientRect(hwnd, &rect);
            cxClient = rect.right;
            cyClient = rect.bottom;
        hdcBackBuffer = CreateCompatibleDC(NULL);
            HDC hdc = GetDC(hwnd);
        hBitmap = CreateCompatibleBitmap(hdc,
                                    cxClient,
                                    cyClient);
            ReleaseDC(hwnd, hdc);
                hOldBitmap = (HBITMAP)SelectObject
(hdcBackBuffer, hBitmap);
    }
    break;
case WM_KEYUP:
    {
        switch(wparam)
        {
        case VK_RETURN:
            {
                g_pPSO->Run(hwnd);
            }
            break;
        case VK_ESCAPE:
            {
                PostQuitMessage(0);
            }
            break;
        case VK_SPACE:
            {
                g_pPSO->Stop();
                }
                break;
            }
        }
        break;
    case WM_SIZE:
        {
            cxClient = LOWORD(lparam);
            cyClient = HIWORD(lparam);
            SelectObject(hdcBackBuffer, hOldBitmap);
            HDC hdc = GetDC(hwnd);
            hBitmap = CreateCompatibleBitmap(hdc,
                cxClient,
                cyClient);
            ReleaseDC(hwnd, hdc);
                hOldBitmap = (HBITMAP)SelectObject
(hdcBackBuffer, hBitmap);
            }
            break;
        case WM_PAINT:
            {
                hdc = BeginPaint(hwnd, &ps);
                BitBlt(hdcBackBuffer, 0, 0, cxClient, cyClient,
NULL, NULL, NULL, WHITENESS);
        g_pPSO->Render(cxClient,cyClient,hdcBackBuffer);
                BitBlt (hdc, 0, 0, cxClient, cyClient,
hdcBackBuffer, 0, 0, SRCCOPY);
                ReleaseDC(hwnd, hdc);
                EndPaint(hwnd, &ps);
            }
            break;
        case WM_DESTROY:
            {
                SelectObject(hdcBackBuffer, hOldBitmap);
              DeleteDC(hdcBackBuffer);
                DeleteObject(hBitmap);
                delete g_pPSO;
                PostQuitMessage(0);
            }
            break;
    default:break;
    }
    return (DefWindowProc(hwnd, msg, wparam, lparam));
```

```
}
//////////////////////////////////////////////////////////////////
int WINAPI WinMain(     HINSTANCE hinstance,
                        HINSTANCE hprevinstance,
                        LPSTR lpcmdline,
                        int ncmdshow)
{
     WNDCLASSEX winclass;
     HWND  hwnd;
     MSG  msg;
     winclass.cbSize= sizeof(WNDCLASSEX);
     winclass.style=CS_HREDRAW | CS_VREDRAW;
     winclass.lpfnWndProc= WindowProc;
     winclass.cbClsExtra  = 0;
     winclass.cbWndExtra = 0;
     winclass.hInstance = hinstance;
winclass.hIcon  = LoadIcon(NULL, IDI_APPLICATION);
     winclass.hCursor= LoadCursor(NULL, IDC_ARROW);
     winclass.hbrBackground = NULL;
     winclass.lpszMenuName = NULL;
winclass.lpszClassName = szWindowClassName;
winclass.hIconSm=LoadIcon(NULL, IDI_APPLICATION);
     if (!  RegisterClassEx(&winclass))
          return 0;
     if (!  (hwnd = CreateWindowEx(NULL,
                        szWindowClassName,
                        szApplicationName,
                  WS_OVERLAPPEDWINDOW | WS_VISIBLE,
                                        0,0,
                              WINDOW_WIDTH,
WINDOW_HEIGHT,
                        NULL,
     NULL,
                        hinstance,
          NULL)))
          return 0;
     ShowWindow(hwnd, ncmdshow);
     UpdateWindow(hwnd);
     bool bDone = false;
     while(!  bDone)
     {
          while  (  PeekMessage  (  &msg,  NULL,  0,  0,
PM_REMOVE ) )
          {
               if( msg.message == WM_QUIT )
               {
                    bDone = true;
               }
               else
               {
                    TranslateMessage( &msg );
                    DispatchMessage( &msg );
               }
          }
          //算法开始运行
          if (g_pPSO->Started())
          {
               g_pPSO->Epoch();
               InvalidateRect(hwnd, NULL, TRUE);
               UpdateWindow(hwnd);
          }
          }
     UnregisterClass  (  szWindowClassName,  winclass.
hInstance );
     return 0;
     }
//////////////////////////////////////////////////////////////////
//
//          文件: defines.h
//          描述: 常用变量和粒子群算法的参数
//
//////////////////////////////////////////////////////////////////
#ifndef DEFINES_H
#define DEFINES_H
//定义窗口大小的长和宽
#define WINDOW_WIDTH  600
#define WINDOW_HEIGHT  600
//定义迷宫大小的长和宽
#define MAP_WIDTH  15
#define MAP_HEIGHT  18
//定义粒子种群大小
#define POP_SIZE  60
//定义粒子的长度及算法参数
#define PARTICLE_LENGTH 35
#define W  1.3
#define C1  1.8
#define C2  1.2
#endif
```

4 结语

PSO 算法与其他的启发式搜索和 A* 算法有很大不同，每次运行程序，它可以找出多条通路，算法运算速度也比较快，实现简单，对于求解迷宫问题有较强的优势。相比于其他算法（如 A* 算法），PSO 算法设计主要的难点在于编码的设计及选择恰当粒子更新所需的参数，既便如此，PSO 仍不失为一种优秀的算法。

（收稿日期：2012-11-05）

基于Java的表达式计算器设计与实现

杨 东

摘 要：介绍了用Java设计与实现表达式计算器的方法。以堆栈的方式，对输入的表达式进行判断与解析，对于符合运算规则的表达式进行计算并输出结果。实现了加、减、乘、除四则运算的功能，并能通过添加括号来改变运算的先后顺序，从而得到预期的运算结果。

关键词：Java语言；表达式解析；栈；计算器

1 引言

在日常的学习与工作过程中，经常遇到计算表达式数值的问题，诸如a-（b+c*d）等，如果人为地分步计算，难免因疏漏了括号或字符而导致结果出错。表达式计算器具有计算表达式结果的功能，大大方便了学习与工作。下面就表达式计算器的设计与实现展开了讨论。

2 设计思想

表达式计算器需实现数据的四则运算功能以及括号与四则运算的优先级问题，即应该先算括号内的内容，后算乘除法，再算加减法。用户通过界面输入待计算的表达式，单击“=”项将获得计算结果，若结果有误，将输出错误提示信息，以便告知用户检查表达式的正确性，修改输入的表达式内容。

3 Java中的栈（Stack）

作为现今主流的编程语言之一，Java提供了栈（Stack）这个类，在栈的定义中提供了如表1所示将会使用到的方法。

表1 Stack类中的方法

方法	说明
public E push (E item)	将对象item压入栈顶
public synchronized E pop ()	栈顶对象出栈
public synchronized E peek ()	查看栈顶对象而不移除该对象
public boolean empty ()	判断栈是否为空

pop () 和peek () 方法返回的是Object，在使用时，可使用toSting () 方法将其转换为在表达式计算过程中的用到的String类型。由于栈的定义位于“Stack.java”中，在使用Stack时，需添加引用“import java.util.Stack;”。

4 开发环境

开发环境：NetBeans IDE 7.0.1；
jdk版本：1.6。

5 表达式解析

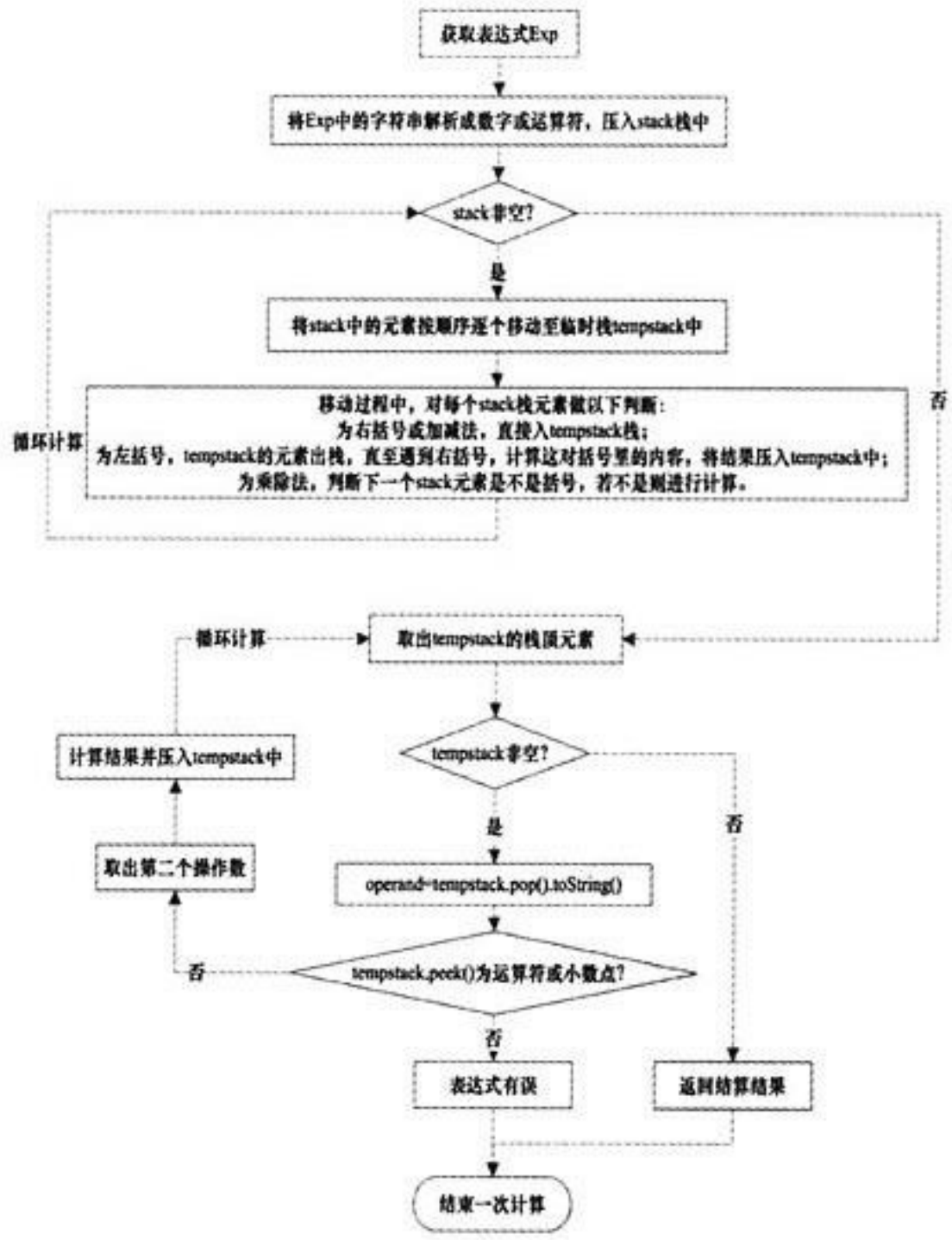


图1 表达式解析流程

对输入的表达式进行解析是计算结果的关键步骤。解析的过程可分两步进行：第一步，对表达式Exp中的字符进行解析，

分离出数字、运算符、括号，按顺序压入 stack 栈中；第二步，将 stack 中的元素往 tempstack 栈中转移，每次移动一个元素，根据元素的类型做出相应处理，待 stack 为空时，tempstack 中将保存最终计算结果。表达式解析的流程如图 1 所示。

6 Calculator 类

Calculator 类实现了输入表达式的解析与结果的计算，是计算器的核心部分。在 Calculator 类中，Cal (String Exp) 方法将参数 Exp 使用堆栈的方法进行解析，按照图 1 中的流程进行分析，如果表达式正确，则返回计算结果，如果表达式是错误的，则输出错误提示。Calculator.java 的代码如下：

```
package calculator;
import java.io.*;
import java.util.Stack;
import java.lang.Character;
public class Calculator  {
    public static String Expression;//记录输入的表达式
    public static int ExpError=0;
    public static double Cal(String Exp)
    {
        double result;
        double tempRes=0;//临时结果
        Stack stack= new Stack();//表达式栈
        Stack tempstack=new Stack();//临时栈
        int index=0;//字符串索引
        String op="";//存放内容
        String op1="";//操作数 1
        String op2="";//操作数 2
        String operand="";//运算符
        //先将 Exp 按照顺序压入栈中
        while(index < Exp.length())
        {
            op+=Exp.charAt(index);
            if(index==Exp.length()-1)
            {//最后一个符号
                stack.push(op);
                op="";
            }
            else if(op.equals("+")||op.equals("-")||op.equals("*")||
        op.equals("/")||op.equals("(")||op.equals(")"))
                {//运算符或括号入栈入栈
                    stack.push(op);
                    op="";
                }
                else if(! Character.isDigit(Exp.charAt(index+1)) &&
        Exp.charAt(index+1)! =′.′)
                {//下一个字符是非数字,非小数点
                    stack.push(op);
                    op="";
                }
            index++;
        }
        //将 stack 中的内容逐渐往 tempstack 中移动,移动过程
    //中进行计算
        while(stack.isEmpty()==false)
        {//stack 不空时进行移动
            if(ExpError==1)
            {
                break;//表达式错误则退出循环
            }
            op=stack.pop().toString();
            if(op.equals(")"))
            {//右括号直接入栈
                tempstack.push(op);
            }
            else if(op.equals("("))
            {//左括号进行计算
                while(! tempstack.peek().equals(")"))//括号匹配
                {
                    op1=tempstack.pop().toString();//从临时栈中取
        //第一个操作数
                    if(tempstack.peek().equals(")"))
                    {
                        tempstack.pop();
                        if(tempstack.isEmpty()==false)
                        {
                            if (tempstack.peek ().equals ("*") ||tempstack.peek ().
        equals("/"))//括号后有乘或者除的情况,先算乘除
                            {
                                operand=tempstack.pop().toString();
                                op2=tempstack.pop().toString();//从栈中取第二个操作数
                                double dop2=Double.parseDouble(op2);
                                double dop1=Double.parseDouble(op1);
                                if(operand.equals("*"))
                                {
                                    tempRes=dop1*dop2;
                                    tempstack.push(tempRes);//本次结果保存到 tempstack 中
                                }
                                else if(operand.equals("/"))
                                {
                                    tempRes=dop1/dop2;
                                    tempstack.push(tempRes);//本次结
                        //果保存到 tempstack 中
                                }
                                op2="";
                                op1="";
                                operand="";
                            }
                            else
                            {
                                tempstack.push(op1);
                            }
```

```
                break;
            }
            else
            {
                tempstack.push(op1);
                break;
            }
        }
        else  if (tempstack.isEmpty () ||op1.equals ("+") ||
op1.equals("-")||op1.equals("*")||op1.equals("/")||op1.equals("."))
            {//栈为空,括号不匹配,或操作数为运算符
                ExpError=1;//表达式错误
                break;
            }
            operand=tempstack.pop().toString();
            //括号内只有一个数,未作出判断
    op2=tempstack.pop().toString();//从栈中取第二个操作数
            double dop2=Double.parseDouble(op2);
            double dop1=Double.parseDouble(op1);
            if(operand.equals("+"))
            {
                tempRes=dop1+dop2;
                tempstack.push(tempRes);//本次结果保存
//到 tempstack 中
            }
            else if(operand.equals("-"))
            {
                tempRes=dop1-dop2;
                tempstack.push(tempRes);//本次结果保存
//到 tempstack 中
            }
            op2="";
            op1="";
            operand="";
        }
        op="";
    }
    else if(op.equals("*") ||op.equals("/"))
    {//乘除先运算
        op2=tempstack.pop().toString();//从临时栈中取第
//二个操作数
        if(stack.isEmpty())
    {//栈空,表达式错误
        ExpError=1;//表达式错误
        break;
    }
    if(stack.peek().equals(")"))
    {//判断 stack 中的下一个符号是否为右括号
        tempstack.push(op2);//乘除遇到括号先不算
        tempstack.push(op);
        continue;
    }
        op1=stack.pop().toString();//从栈中取第一个操作数
            double dop2=Double.parseDouble(op2);
            double dop1=Double.parseDouble(op1);
            if(op.equals("*"))
            {
                tempRes=dop1*dop2;
                tempstack.push(tempRes);//本次结果保存到
//tempstack 中
            }
            else
            {
                tempRes=dop1/dop2;
                tempstack.push(tempRes);//本次结果保存到
//tempstack 中
            }
            op2="";
            op1="";
        }
        else if(op.equals("+" ) ||op.equals("-"))
        {//加减先不运算,直接移动到 tempstack 中
            tempstack.push(op);
        }
        else//为数字
        {
            if(stack.isEmpty())
            {//栈为空进行计算
                tempstack.push(op);
                while(tempstack.isEmpty()==false)
                {//临时栈有数据
                    try
                    {
                        op1=tempstack.pop().toString();
                        if(tempstack.isEmpty())
                        {
                tempstack.push(op1);//最后一个元素,不处理
                            break;
                        }
                        operand=tempstack.pop().toString();
                        if(tempstack.isEmpty()||op1.equals("+")||
op1.equals("-")||op1.equals("*")||op1.equals("/")||op1.equals("."))
                        {//栈内已无元素,表达式有错误
                            ExpError=1;//表达式错误
                            break;
                        }
                        op2=tempstack.pop().toString();
                        double dop2=Double.parseDouble(op2);
                        double dop1=Double.parseDouble(op1);
                        if(operand.endsWith("+"))
                        {
                            tempRes=dop1+dop2;
                        }
                        else if(operand.equals("-"))
```

```
        {
            tempRes=dop1-dop2;
        }
        tempstack.push(tempRes);//结果入栈
        op1="";
        op2="";
        operand="";
    }
    catch(Exception e)
    {
        break;//退出循环
    }
    }
    }
    else
    {//栈非空,寻找括号或者其他可以计算的标志
        tempstack.push(op);
    }
    }
    op="";//op 重置
}
//上面表达式分解完毕,以下判断栈内是否还有加减未进
//行运算的
    while(tempstack.isEmpty()==false)
    {//临时栈有数据
        try
        {
            op1=tempstack.pop().toString();
            if(tempstack.isEmpty())
            {
                tempstack.push(op1);//最后一个元素,不处理
                break;
            }
            operand=tempstack.pop().toString();
            if(tempstack.isEmpty()||op1.equals("+")||op1.equals
("-")||op1.equals("*")||op1.equals("/")||op1.equals("."))
            {//栈内已无元素,表达式有错误
                ExpError=1;//表达式错误
                break;
            }
            op2=tempstack.pop().toString();
            double dop2=Double.parseDouble(op2);
            double dop1=Double.parseDouble(op1);
            if(operand.endsWith("+"))
            {
                tempRes=dop1+dop2;
            }
            else if(operand.equals("-"))
            {
                tempRes=dop1-dop2;
            }
            tempstack.push(tempRes);//结果入栈
            op1="";
            op2="";
            operand="";
        }
        catch(Exception e)
        {
            break;//退出循环
        }
    }
    result=Double.parseDouble(tempstack.pop().toString());
    return result;
  }
}
```

7 CalJFrame 类

CalJFrame 类实现了窗体的设计与表达式的生成，使用 netBeans 提供的组件面板添加了 btn0-btn9 这 10 个数字按钮以及加、减、乘、除、左括号、右括号、退格、清零、等于的按钮后，为数字 0-9 的按钮添加统一的 btnNumClicked 事件，根据按下的按钮判断表达式中添加的是哪一个数字。该部分主要代码如下：

```
package calculator;
import java.awt.Color;
public class CalJFrame extends javax.swing.JFrame {
    public static int isStart=1;//是否为新运算
    public static Color btnColor;//按钮颜色
    public CalJFrame() {
        initComponents();
        setLocationRelativeTo(this);//居中显示
      }
      @SuppressWarnings("unchecked")
       private void initComponents(){...}//窗体设计的代码,由设
//计器生成
public void IsStart()
{
    if(isStart==1)
    {//新运算
        txtExpression.setText("");//清空原有内容
        txtResult.setText("");
        isStart=0;
    }
}
private void btnNumClicked(java.awt.event.ActionEvent evt) {
// 数字键 0-9 被按下
    //判断是否为一次新的计算
    IsStart();
    String Num=evt.getActionCommand();
    txtExpression.setText(txtExpression.getText()+Num);
}
private    void    btnDoteActionPerformed    (java.awt.event.
```

```
ActionEvent evt) {
// 小数点被按下
    IsStart();
    if(! txtExpression.getText().isEmpty())
    {//表达式非空
        txtExpression.setText(txtExpression.getText()+".");
    }
}
private void btnLeftActionPerformed (java.awt.event.
ActionEvent evt) {
// 左括号被按下
    IsStart();
    txtExpression.setText(txtExpression.getText()+"(");
}
private void btnRightActionPerformed (java.awt.event.
ActionEvent evt) {
// 右括号被按下
    IsStart();
    txtExpression.setText(txtExpression.getText()+")");
}
private void btnOperandClicked (java.awt.event.ActionEvent
evt) {
// 运算符被按下
    IsStart();
    if(! txtExpression.getText().isEmpty())
    {//表达式非空
        String Operand=evt.getActionCommand();
  txtExpression.setText(txtExpression.getText()+Operand);
    }
}
private void btnBackActionPerformed (java.awt.event.
ActionEvent evt) {
// 退格一次
    IsStart();
    if(! txtExpression.getText().isEmpty())
    {//表达式非空
        String express1=txtExpression.getText();
        String express2=express1.substring(0, express1.length
()-1);//去掉最后一个字符
        txtExpression.setText(express2);//显示新表达式
    }
}
private void btnClearActionPerformed (java.awt.event.
ActionEvent evt) {
// 清空按钮
    txtExpression.setText("");//清空文本内容
    txtResult.setText("");
}
private void btnResultActionPerformed (java.awt.event.
ActionEvent evt) {
// 等于按钮按下,求值处理
    try
    {
        Calculator.ExpError=0;//清除错误记录
double result=Calculator.Cal(txtExpression.getText().trim());
        if(Calculator.ExpError==0)
        {
            txtResult.setText(Double.toString(result));//显示结果
            isStart=1;//开始新的计算
        }
        else
        {//表达式错误
            txtResult.setText("Expression Error ");//显示错误提示
            isStart=1;//开始新的计算
        }
    }
    catch(Exception e)
    {
        txtResult.setText("Expression Error ");//显示错误提示
        isStart=1;//开始新的计算
    }
}
private void btnFocus(java.awt.event.MouseEvent evt) {
// 鼠标移动至按钮上
    btnColor=btn0.getBackground();
    evt.getComponent().setBackground(Color.orange);
}
private void btnNotFocus(java.awt.event.MouseEvent evt) {
// 鼠标离开按钮
    evt.getComponent().setBackground(btnColor);//设置回原来
//的颜色
}
    public static void main(String args[]) {
        /* Set the Nimbus look and feel */
        try {
            for   (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel (info.
getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
            java.util.logging.Logger.getLogger (CalJFrame.class.
getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (InstantiationException ex) {
            java.util.logging.Logger.getLogger (CalJFrame.class.
getName()).log(java.util.logging.Level.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {
            java.util.logging.Logger.getLogger (CalJFrame.class.
getName()).log(java.util.logging.Level.SEVERE, null, ex);
                                                } catch (javax.swing.
```

(下转第 21 页)

PDF 文档的活动书签

徐怀平

摘　要：PDF 文档的通用性不容质疑，不过如果经常在不同的电脑上看 U 盘或是移动盘上的电子书时，肯定困惑于书签的问题，许多工具的书签都记录在注册表中，给动态看电子书的读者带来一定麻烦，而以文件形式记录电子书签以及附加书签到文件名上的形式，既方便在不同电脑上阅读的连续性，又便于在移动设备上自行定位。

关键词：PDF 文档；书签；移动；AU3 工具

1　需求来源

平常看电子书，可能会在不同的电脑上进行看，而且许多 PDF 文档也会在移动设备上看看，如何让看书的位置方便记忆是一个大问题。因为长篇幅的 PDF 电子书可不是一时三刻就能够看完的。

2　实现目标

要想让电子书能够方便记住位置，可以：

（1） 将阅读器对 PDF 文件阅读记录到一个列表中。

（2） 如果要让移动设备上也方便查看，可以把页码记录到文件名中，便于手工定位。

（3） 提供直接打开最近一次的 PDF 文档的阅读。

3　面临问题

工具基于 foxit reader 来完成书签位置的提取和指定页面的打开，由于工具是一个变化发展的过程，原来在低版本支持的功能，也许是用的人少，某些命令行功能到了高版本反而得不到支持，因此在处理时需要注意当前装载的 foxit pdf reader 的版本问题，在保存列表、打开文档时针对不同的版本要进行不同的处理。

4　实现过程

4.1　命令行的处理

在实现小工具时，需要注意程序的两种模式，一种是命令行功能的支持，另外一种是打开界面的处理，根据传入命令行的个数来决定，目前采用：

（1） 打开 PDF 文件，注意会根据文件名和文件信息列表决定书签位置：Pdftool pdffile。

（2） 同步 PDF 文件的当前页面信息到文件：pdftool －sam pdffile。

（3） 打开工具命令行界面：pdftool。

实际使用时，命令行其实是集成到系统中的，对于用户来说是不可见的，操作可视化，具体内容如下：

```
$li_cmd = $cmdline[0]
Switch $li_cmd
    Case 1
        $ls_filename = $cmdline[1]
    Case 2
        $ls_cmd = $cmdline[1]
        $ls_filename = $cmdline[2]
EndSwitch
;必须有 2 个参数 分别表示目录名称和相应的命令模式
If $li_cmd > 0 Then
    If $li_cmd >1 Then
        If StringCompare($ls_cmd, "sam")=0 Then
            PdfSamePage( $ls_filename )
            Exit
        EndIf
        Exit
    EndIf
    If Not FileExists( $ls_filename ) Then
MsgBox(64,"目录不存在", "指定的目录或者是文件不存在！")
        Exit
    EndIf
    ;如果是 PDF 文件由进行相应的定位
    $newname = $ls_filename
    $ls_ext = fs_extname( $ls_filename )
    If StringCompare($ls_ext, "pdf")=0 Then
        pdfOpen( $newname )
    EndIf
    Exit
EndIf
GUICreate( $ls_toolname, 380, 200 )
$But_back = GUICtrlCreateButton("备份当前 PDF 文档的书签", 40, 40, 300, 35, $WS_GROUP)
$But_open = GUICtrlCreateButton("自动打开刚阅读的 PDF
```

```
文档", 40, 90, 300, 35, $WS_GROUP)
  $But_reg = GUICtrlCreateButton (" 注册加书签功能", 40,
140, 300, 35, $WS_GROUP)
  GUISetState(@SW_SHOW)
  While 1
     $nMsg = GUIGetMsg()
     Switch $nMsg
     Case -3
          Exit
          ;书签备份
    Case $But_back
          pdfBack()
          exit 0
    Case $But_reg
          pdfReg()
          Exit
    Case $But_open
          pdfOpenRecent()
          Exit 0
    EndSwitch
  WEnd
  Exit
```

4.2 同步功能的注册与实现

对于同步书签到文件名来说，使用“pdftool sam pdffile”，实际上实现时关联到 PDF 文件类型来完成（函数 pdfReg）：

```
    #pdf, 需要注意用户的选项
    $s_root = "HKEY_CLASSES_ROOT\"
    $s_type = RegRead( $s_root&".pdf", ".ksobak")
    If StringLen($s_type)=0 Then
         $s_type = RegRead( $s_root&".pdf", "")
    EndIf
    $s_reg = $s_root & $s_type & "\shell\ 同步 PDF 位置
(&A)\command"
    $s_run = chr(34) & @ScriptFullPath & chr(34) & " sam "
& chr(34) & "%1" &chr(34)
    $i_ret = RegWrite($s_reg, "", "REG_SZ", $s_run)
    Return 1
```

利用系统的注册表功能，找到 PDF 文件关联到的文件类型，提供“同步 PDF 位置”的功能，其中“&A”提供的是快捷键功能，可以根据需要设定成其他不冲突的按键，如图 1 所示。

图 1 添加快捷键

相对来说同步页面功能也并不复杂（PdfSamePage）：

```
Func PdfSamePage( $pdfFile )
   $ls_key = "HKCU\Software\Foxit Software\ 福昕 PDF 阅
读器 5.1\Recent File List"
      $ls_file = RegRead($ls_key, "File1" )
      If StringLen($ls_file)>0 Then
            pdfSame51( $pdfFile )
            Return "OK"
      EndIf
      $ls_key = "HKCU\Software\Foxit Software\ 福昕阅读器
\Recent File List"
      $ls_file = RegRead($ls_key, "File1" )
      If StringLen($ls_file)>0 Then
            pdfSame50( $pdfFile )
            Return "OK"
      EndIf
      Exit
  EndFunc
  Func pdfSame51( $pdfFile )
      $li_max = RegRead ( "HKCU\Software\Foxit Software\
福昕 PDF 阅读器 5.1\Preferences\History\Options", "
nSaveMaxSize")
      $as_key = "HKCU\Software\Foxit Software\ 福昕 PDF 阅
读器 5.1\Preferences\History\LastOpen"
      ;MsgBox(64, "51", $as_key )
      pdfSame00( $li_max, $as_key, $pdfFile )
      Return $li_max
  EndFunc
  Func pdfSame50( $pdfFile )
  $as_key = "HKCU\Software\Foxit Software\ 福昕阅读器 \"
     ;取出最大的文件数
     $ls_maxkey = $as_key & "ChildFrame"
     $li_max =  RegRead( $ls_maxkey, "HistoryMaxSize")
     $as_key = $as_key & "\History"
     pdfSame00( $li_max, $as_key, $pdfFile )
     Return $li_max
  EndFunc
  ;2012-03-08 根据传入的文件和页码生成新的文件名
  Func pdfNewName( $pdfFile, $newPage )
     $ls_name = fs_filename( $pdfFile )
     $ls_path = Stringleft ( $pdfFile, StringLen ($pdfFile) -
StringLen($ls_name) )
     $i = StringLen($ls_name)
     While $i>0
          $ls_char = StringMid( $ls_name, $i, 1 )
          If StringCompare($ls_char, "-")=0 Then
$ls_retname = StringLeft( $ls_name, $i ) & $newPage & ".pdf"
               Return $ls_path & $ls_retname
          EndIf
          $i = $i - 1
     WEnd
     $ls_retname = StringLeft ( $ls_name, StringLen
($ls_name) - 4) & "-" & $newPage & ".pdf"
     Return $ls_path & $ls_retname
  EndFunc
```

```
;2012-03-08 同步相应文件的书签信息
Func pdfSame00( $li_max, $as_key, $pdfFile )
    $ls_info = ""
    $ls_pageinfo = ""
    $ls_pdfFile = StringUpper( $pdfFile )
    For $i=1 To $li_max Step 1
    $ls_filename = RegRead($as_key & "\" & $i, "FileName")
    $ls_readpage = RegRead($as_key & "\" & $i, "page")
    $ls_readscale = RegRead($as_key & "\" & $i, "Scale")
        If FileExists( $ls_filename) Then
    $ls_newname = FileGetLongName( $ls_filename )
            $ls_newname = StringUpper( $ls_newname )
    If StringCompare( $ls_pdfFile, $ls_newname ) = 0 Then
    $ls_newPdf = pdfNewName( $ls_pdfFile, $ls_readpage)
                FileMove( $ls_pdfFile, $ls_newPdf )
        $ls_newname = FileGetShortName( $ls_NewPdf )
                RegWrite ( $as_key & "\" & $i, "
FileName", "REG_SZ", $ls_newname )
                Exit
            EndIf
        EndIf
    Next
EndFunc
```

可以看出为了实现该功能就必须考虑到不同版本的问题，不同版本使用的注册表位置不同，首先取得相应版本的文件列表位置，取得文件名，根据需要把文件名改为带有书签的文件名，由于工具保存的是短文件名，还需要使用FileGetShortName取得相应的短文件名，替换原来保留的文件名，这样当在本环境中继续打开该文档时仍会使用新文件名来延续相应的文件位置的功能。

4.3 打开 PDF 并定位页码

在打开 PDF 文档时，要注意两种情况，一种是把文档的页码附加到文件上，另外一种是把页码保存在文档列表中的，因此针对不同情况进行不同处理：

```
Func pdfOpen( $pdfFile )
    ;加载已经存在的信息
    pdfLoadPage()
    $ls_pdfexe = pdfExe()
    $ls_page = pdfPage( $pdfFile )
    ;取出最大的文件数
    $ls_key = ""
    $li_max = RegRead ( "HKCU\Software\Foxit Software\
福昕 PDF 阅读器 5.1\Preferences\History\Options", "
nSaveMaxSize")
    If $li_max>0 Then
        $ls_key = "HKCU\Software\Foxit Software\ 福昕
PDF 阅读器 5.1\Preferences\History\LastOpen"
        $ls_shortname = FileGetShortName( $pdfFile )
        $li_max = RegRead ( "HKCU\Software\Foxit
Software\ 福昕 PDF 阅读器 5.1\Preferences\History\Options",
"nSaveMaxSize")
        For $i=1 To $li_max Step 1
    $ls_filename = RegRead($ls_key & "\" & $i, "FileName")
    $ls_readpage = RegRead($ls_key & "\" & $i, "page")
        Next
        If StringLen($ls_page)>0 Then
       RegWrite ( $ls_key & "\1", "FileName", "REG_SZ",
$ls_shortname )
            $li_pos = StringInStr( $ls_page, "\" )
            $ls_scale = ""
            If $li_pos>0 Then
        $ls_scale = StringMid( $ls_page, $li_pos + 1 )
        $ls_page = StringLeft( $ls_page, $li_pos – 1 )
            EndIf
    RegWrite( $ls_key & "\1", "Page", "REG_SZ", $ls_page )
            ;2012-01-06 显示比例
            If StringLen($ls_scale)>0 Then
 RegWrite( $ls_key & "\1", "Scale", "REG_SZ", $ls_scale )
            EndIf
        EndIf
$run = $ls_pdfexe & " " & Chr(34)  & $pdfFile & Chr(34)
        Sleep(2000)
        Run( $run )
        Exit
    EndIf
    If StringLen($ls_page)>0 Then
        $ls_page = "–n "& $ls_page
    EndIf
        $run = $ls_pdfexe & " " & Chr (34)  & $pdfFile &
Chr(34) & " " & $ls_page
    Run( $run )
EndFunc
```

如果是 4.3 版本是可以通过命令行参数的方式把页码直接传送过去的，而对于 5.1 版本则对命令行不再支持，只能修改注册表中的文件信息并且在打开阅读器之后才能生效，因此对于 5.1 版则存在一个问题，如果相应的阅读器已经打开时，可能页码跳转功能并不能生效。

在处理文件页码函数 pdfLoadPage 则是读取保存下来的文件列表信息，在 pdfPage 中完成对页码的处理，如果在文件名中已经包含了页码，则返回此页码；如果不存在，则取文件列表中对应的页码返回到程序中。

如果不希望工具来干涉页码时，则可以直接双击相应的 PDF 文档，而不是把它拖放到工具上。

4.4 最近阅读的文档

对于最近打开的 PDF 文档，则相对简单，在函数pdfOpenRecent 中处理，首先使用 pdfReadkey 来取得不同版本的列表位置：

```
;2012-01-02 要注意新版本有变化, 5.1 版键值有变化
Func pdfReadkey( )
    $ls_key = "HKCU\Software\Foxit Software\ 福昕 PDF 阅
```

```
读器 5.1\Recent File List"
    $ls_file = RegRead($ls_key, "File1" )
    If StringLen($ls_file)>0 Then
        Return $ls_key
    EndIf
    $ls_key = "HKCU\Software\Foxit Software\ 福昕阅读器
\Recent File List"
    $ls_file = RegRead($ls_key, "File1" )
    If StringLen($ls_file)>0 Then
        Return $ls_key
    EndIf
    Return ""
EndFunc
```

之后利用以下语句：

```
$ls_file = RegRead($ls_key, "File1" )
```

取得对应的首文件名，再打开阅读即可。

5 结语

工具看起来并不大，但是它的实用性却很强，把文件列表放入网络磁盘（快盘等工具）中之后，一旦文件阅读到中途，想更换电脑或者是在 pad 中继续阅读时，可以把页码放置到文件名中，这样可以通过在 pad 中手动跳转页码。

而在 pad 中阅读后，也可以手动在文件名上修改页码同步到电脑中，在电脑上使用工具打开到指定页码，保证阅读的连贯性。如果放置到移动硬盘上看时，同步好页码，阅读也同样体现了它的连续性。

（收稿日期：2012-09-26）

（上接第 17 页）

```
UnsupportedLookAndFeelException ex) {
        java.util.logging.Logger.getLogger (CalJFrame.class.
getName()).log(java.util.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new CalJFrame().setVisible(true);
        }
    });
}
// Variables declaration - do not modify
private javax.swing.JButton btn0;
private javax.swing.JButton btn1;
private javax.swing.JButton btn2;
private javax.swing.JButton btn3;
private javax.swing.JButton btn4;
private javax.swing.JButton btn5;
private javax.swing.JButton btn6;
private javax.swing.JButton btn7;
private javax.swing.JButton btn8;
private javax.swing.JButton btn9;
private javax.swing.JButton btnAdd;
private javax.swing.JButton btnBack;
private javax.swing.JButton btnClear;
private javax.swing.JButton btnDiv;
private javax.swing.JButton btnDote;
private javax.swing.JButton btnLeft;
private javax.swing.JButton btnMin;
private javax.swing.JButton btnMul;
private javax.swing.JButton btnResult;
private javax.swing.JButton btnRight;
private javax.swing.JPanel jPanel1;
private javax.swing.JTextField txtExpression;
private javax.swing.JTextField txtResult;
// End of variables declaration
}
```

8 表达式计算器的运行

运行表达式计算器后，输入正确的运算表达式，将得到运算结果见图 2。若输入错误，显示错误提示，如图 3 所示。

图 2 计算结果

图 3 表达式错误提示

9 结语

通过使用堆栈的方式，设计与实现了表达式计算器。在表达式的解析过程中，仅对基本的四则运算以及括号进行了分析，没有科学计算的功能，如有需要，可以添加数学函数的功能实现科学计算。

参考文献

[1] 张永常，主编. Java 程序设计实用教程 [M] . 北京：电子工业出版社，2006.

[2] 雍俊海. Java 程序设计教程 [M] . 北京：清华大学出版社，2007.

（收稿日期：2012-09-04）

Flash as3 连线题制作类的设计、实现与应用

程海生

摘　要：连线题作为传统题型的重要成员有其自身的特点和优势，在电子测试系统中利用 Flash 制作连线题比较繁琐，为此编写了 Flash 连线题制作程序并封装成一个 as3 类，有了这个类制作连线题就变得非常简单了，不太懂 as3 编程的人也能轻松完成，而且功能完善，有效地延展了 Flash 制作连线题的功能，提高了教师制作 Flash 连线题的效率。

关键词：Flash 软件；连线题；as3 类；快速；简单实用；功能完善

1　引言

在多媒体网络教学环境中常见的题型有单选题、多选题、填空题，纸质试卷中常见的连线题在网络环境中就不多见了，以前也有人用 Flash 制作连线，但制作起来比较繁锁，每次制作都需要编写程序造成大量的重复劳动，为此编写了一个 Flash as3 连线题制作类，使得 Flash 连线题的制作变得非常简单、快速。

2　Flash as3 连线题制作类的设计

连线题制作类的设计目的是要将连线题的全部功能由一个类来现实，只要有了这个类文件教师就能轻松制作连线题，尽可能的减少教师 Flash 连线题课件的大量重复劳动。大体思路是制作连线题时，创建一个基类为“lianxian.as”（连线题制作类）的影片剪辑元件，使用者在其中自由绘制自己喜欢的图形作为连线按钮使用，然后在舞台上根据问题创建多个实例，只需对应的连线两端的实例名一致即可，这样就能完成了连线题的制作，成绩也要自动统计保存在变量中。

3　as3 连线题制作类的应用

连线题

1、5+9=　　13

2、6+7=　　11

3、3+8=　　14

4、6+6=　　15

5、8+7=　　12

图 1　设置连线问题

在 Flash cs4（Flash cs3.0 以后版本都可以），新建一个“Flash ActionScript 3.0”文件，将其和 lianxian.as（lianxian.as 为编写好的连线题制作类，具体内容在后面）保存在同一个文件夹中。接下来选中第一帧，使用文本工具在舞台上输入问题，如图 1 所示。

然后插入影片剪辑元件，名称为“连线端点”，单击“高级”按钮，选中复选按钮“为 ActionScript 导出 (x)”，并将基类改为“lianxian”，如图 2 所示。

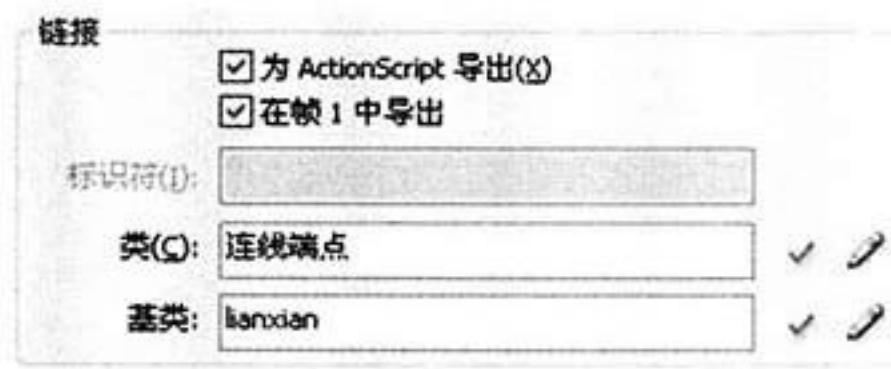


图 2　插入影片剪辑

单击对话框的“确定”按钮，这时出现“类警告”对话框，单击“确定”就可以。然后在“连线端点”影片剪辑中绘制一个图形，这里要注意一定要让图形的中心和注册点（中间的十字）大致对齐。如图 3 所示。

图 3　设置图形的位置

返回主场景，将库中的绘制好的“连线端点”影片剪辑元件，拖动到舞台上，创建元件的实例，并将其大小调整合适，然后复制成多个，并用对齐工具排列整齐。如图 4 所示。

然后在属性面板中为每个“连线端点”元件的实例命名，命名时只需要对应答案的实例名相同即可。如将图 2 所示的左侧 5 个“连线端点”影片剪辑实例，从上到下依次命名为“a”，“b”，“c”，“d”，“e”，根据答案将右侧的自上而

下依次命名为“b”，“c”，“a”，“e”，“d”。

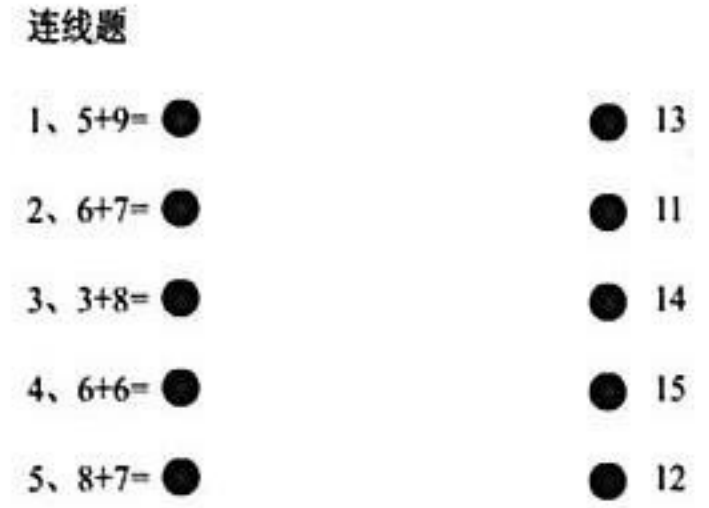


图 4　设置连线端点

至此 Flash 连线题已经可以使用了，单击一个“连线端点”开始画线，再单击另一个停止画线，并且正确的连线条数会保存在变量 a.score 中（其中 a 是基类为“lianxian”的“连线端点”元件的任意一个实例，本例中用 b.score，e.score 等都可以）。

下面再向舞台添加两个按钮和一个动态文本框，把这个 Flash as3 连线题类的全部功能展示出来。首先向舞台添加两个按钮组件，分别把它们的“label”命名为“重做”和“判断”，并将它们的实例名分别命名为“qz”和“pd”。再向舞台添加一个动态文本框，用于判断后显示正确的连线数，在属性面板中为其命名为“chengji”，然后选中时间轴的第一帧，添加如下语句：

```
qz.addEventListener (MouseEvent.CLICK,function (e:MouseEvent){chengji.text="";a.myInit();});//a 为任意一个“连线
//端点”元件的实例名，myInit()函数清除所有画线，并进行初始
//化。
pd.addEventListener (MouseEvent.CLICK,function (e:MouseEvent){a.panduan();chengji.text="正确数:"+a.score;});
//paduan()函数对连线情况进行判断，正确的线由蓝色变为绿
//色，动态文本框"chengji"中显示正确的连线数。
stop();
```

连线题做完后单击判断按钮后的截图如图 5 所示，其中绿色的连线为正确的连线。

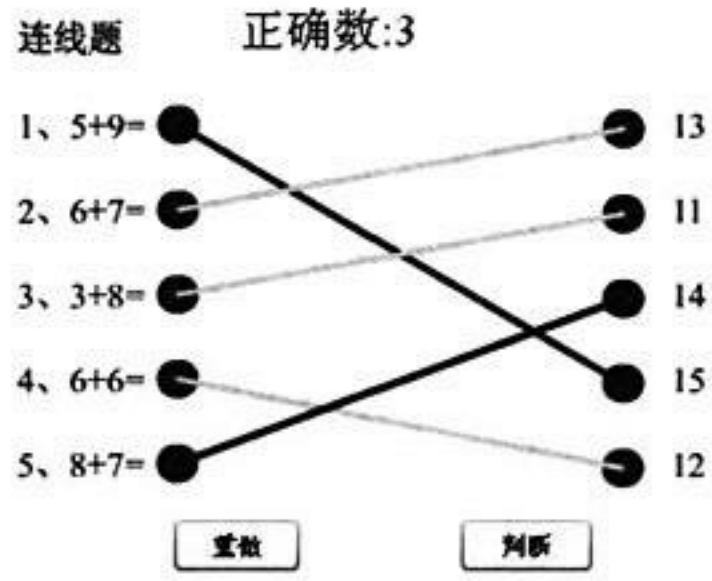


图 5　运行效果

如果需要改变连线的颜色和判断后正确连线的颜色只需添加如下语句：

```
lianxian.prototype.color1="0x990088";//设置连线的颜色，
//可任意设置
lianxian.prototype.color2="0x338800";//设置判断后正确连
//线的颜色
```

4　Flash as3 连线题制作类的具体实现

在 Flash cs4 中新建一个“ActionScript 文件”，输入以下内容，并以“lianxian.as”为文件名存盘。

```
package {
    import flash.display.MovieClip; //导入必要的类
    import flash.events.MouseEvent;
    import flash.events.TimerEvent;
    import flash.utils.Timer;
    public class lianxian extends MovieClip {
        var m_x:Number; //用于保存鼠标坐标
        var m_y:Number;
        var clickFlag=true;//用于是否响应单击操作
        prototype.score=0; //用于保存正确的连线数，是一
//个共享变量
        prototype.drawFlag=true; //共享变量 drawFlag，用
//于标识画线状态还是结束画线状态
        private var myArray=new Array; //定义数组
        prototype.myTimer=new Timer(10); //定义计时器
//共享变量
        prototype.color1="0x0000ff";//共享变量 color1 保存
//连线的颜色
    prototype.color2="0x00ff00";//判断后，正确连线的颜色
        public function lianxian() {//构造函数
    prototype.spr_Arr=myArray; //将数组赋值给共享变量
            this.addEventListener (MouseEvent.CLICK,
drawLine1); //添加鼠标帧听事件
        }
        private function drawLine1(e:MouseEvent) {
            if (clickFlag) {//clickFlag 为 true 时执行程序
                clickFlag=false;
                m_x=mouseX; //保存鼠标坐标
                m_y=mouseY;
                  if (prototype.drawFlag) {//drawFlag 为
//true 时可以画线，否则结束画线
                    prototype.beginName=this.name;
//保存连线起点实例名
                    prototype.spr=new lineSprite;
//mySpriteLine 类后面有定义，用于画线的容器，继承自 Sprite
//类，保存有画线的起点和终点坐标及连线是否正确。
                    prototype.spr.rightFlag=false;
//标记这条连线是否正确
                prototype.spr.x1=this.x;//保存连线起点坐标
                    prototype.spr.y1=this.y;
                     parent.addChild(prototype.spr);
```

```
//将画线在父剪辑中显示出来
                prototype.myTimer.start();//计时器开始
                    prototype.drawFlag=false; //为 false 时,
//为结束画线状态,也就是连线的终点。
                                        prototype.myTimer.
addEventListener(TimerEvent.TIMER,drawLine2);
                    } else {
                prototype.myTimer.stop();//停止画线
                        if    (prototype.beginName  ==this.
name) {//当起点和终点实例名相同时,正确数加 1,并标记这条
//线是正确的
                                prototype.score++;
                                prototype.spr.rightFlag=true;
                            }
                prototype.drawFlag=true;//回到画线状态
                    prototype.spr.x2=this.x;//保存终点坐标
                        prototype.spr.y2=this.y;
                            prototype.spr_Arr [prototype.
spr_Arr.length]=prototype.spr; //将画线容器保存到数组
                prototype.spr.graphics.clear();//清除画线
                            prototype.spr.graphics.lineStyle (3,
prototype.color1,1); //设置线条样式
                                prototype.spr.graphics.moveTo
(prototype.spr.x1,prototype.spr.y1);//设置画线起点
            prototype.spr.graphics.lineTo(this.x,this.y);//画线
                        }
                    }
            }
            private function drawLine2(e:TimerEvent) {
                prototype.spr.graphics.clear();//清除画线
                if (m_x!=mouseX||m_y!=mouseY) {//鼠标移动
//时画线
        prototype.spr.graphics.lineStyle(3, prototype.color1,1);
            prototype.spr.graphics.moveTo    (prototype.spr.x1,
prototype.spr.y1);
                        var addX:int; //连同以下几句使得画线的
//终点距鼠标有一小段距离,以免单击鼠标时,单击到画线上,不
//能响应鼠标事件
                        var addY:int;
                        if (parent.mouseX>prototype.spr.x1) {
                            addX=-2;
                        } else {
                            addX=2;
                        }
                        if (parent.mouseY>prototype.spr.y1) {
                            addY=-2;
                        } else {
                            addY=2;
        }                               prototype.spr.graphics.lineTo
(parent.mouseX+addX,parent.mouseY+addY);
                }
            }
```

```
public function myInit() {//清除连线,进行初始化
        prototype.score=0;
        prototype.spr_Arr.length=0;
        prototype.drawFlag=true;
        for    (var  i:uint =parent.numChildren -1;  i >0;
                                                        i--) {
            if (parent.getChildAt(i) is lineSprite) {
                parent.removeChildAt(i);
            } else if (parent.getChildAt(i) is lianxian) {
                                var  lx:lianxian =parent.getChildAt (i)
as lianxian;
                                lx.clickFlag=true;
                        }
                    }
                }
            public function panduan() {//判断连线对错,统计正
//确连线数,正确的连线绿色显示
                    for   (var i=0; i<=prototype.spr_Arr.length-1;
i++) {
                        if (prototype.spr_Arr[i].rightFlag) {
    prototype.spr_Arr   [i].graphics.lineStyle   (3,  prototype.
color2,1);
prototype.spr_Arr [i].graphics.moveTo (prototype.spr_Arr [i].x1,
prototype.spr_Arr[i].y1);
prototype.spr_Arr   [i].graphics.lineTo   (prototype.spr_Arr [i].x2,
prototype.spr_Arr[i].y2);
                        }
                    }
                }
            }
        }
        import flash.display.Sprite;
        class lineSprite extends Sprite {//自定义 mySpriteLine 容
//器用于画线和保存线条信息
            var rightFlag=false;//连线是否正确
            var x1;//起点坐标
            var y1;
            var x2;//终点坐标
            var y2;
           function lineSprite() {
            }
        }
```

5 结语

介绍了一个用于制作连线题的 Flash as3 类，这个类有效地延展了 Flash as3 制作连线题的功能，避免了制作 Flash 连线题的大量重复工作，效率大大提高，减小了教师制作 Flash 连线题的难度和工作量。

（收稿日期：2012-09-07）

C++Builder 下基于 CppWebBrowser 的“考试系统”设计与实现

黄永兴

摘　要：在 C++Builder 环境下，通过一个实例详细介绍了基于 CppWebBrowser 和 Access2003 设计开发考试系统的方法，包括系统数据结构的设计、HTML 试卷的生成与评判、HTML 页面自动填充等相关问题，为基于数据库开发考试系统提供了一个详实的实例参考。

关键词：CppWebBrowser 组件；Access 2003 数据库；HTML 试卷；考试系统

1　引言

前段时间，单位组织专业技术理论考试，要求开发一款单机版考试系统，并要求操作简单，安装方便，可以支持多个专业同时考试。起初，作者计划选用 Visual Studio 2010 C# 开发，但考虑到其无法脱离 FrameWork 支持环境，发布时安装复杂，故选用 C++ Builder 作为开发平台，数据库选用 Access 2003，以 HTML 网页形式生成试卷，开发出了一款绿色版“专业技术理论考试系统”。不算 mdb 数据库文件，系统本身小于 2MB，在 Windows XP 环境下无需安装即可正常运行。

2　框架设计

在系统编码之前，首先要进行了系统数据库、数据结构和系统流程设计。

2.1　数据库

“专业技术理论考试系统”采用 Access 2003 作为后台数据库，数据库文件命名为 DataBase.mdb，其中包含单位、专业、题型、题库 4 个数据表，用于存储相应信息。各表字段设计如下：

2.1.1 专业类型表（Zy）

专业类型表中主要存储题库中试题的专业划分，系统登录时，从专业下拉列表中选取考试专业，生成相应试卷，如表 1 所示。

表 1　专业类型

字段名称	数据类型	说明
Zy_ID	数字	专业 ID
Zy_Name	文本	专业名称

2.1.2 单位表（Dw）

单位表中主要存储参考单位列表，系统登录时，从单位下拉列表中选取所在单位登录，如表 2 所示。

表 2　单位信息

字段名称	数据类型	说明
Dw_ID	数字	单位 ID
Dw_Name	文本	单位名称

2.1.3 题型表（Tx）

题型表中存储题库中试题类型，系统支持单选题、多选题、判断题、填空题、简答题等类型，如表 3 所示。

表 3　题型

字段名称	数据类型	说明
Tx_ID	数字	题型 ID
Tx_Name	文本	题型名称

2.1.4 专业题库表（Tk）

题库表中按照专业类型、试题类型存储所有的试题，如表 4 所示。

表 4　专业题库

字段名称	数据类型	说明
Tk_ID	自动编号	试题 ID
Zy_ID	数字	专业 ID
Tx_ID	数字	题型 ID
Tg	备注	题干部分
Da	文本	答案部分
Xx1	文本	选项 1
Xx2	文本	选项 2
Xx3	文本	选项 3
Xx4	文本	选项 4

字段名称	数据类型	说明
Xx5	文本	选项 5
Xx6	文本	选项 6
Xx7	文本	选项 7
Xx8	文本	选项 8

2.1.5 数据表关系图

在设计数据库表时，设定" Zy"、" Tx" 与" Tk" 表之间按 Zy_ID 和 Tx_ID 设定对应关系，表之间的关系如图 1 所示。

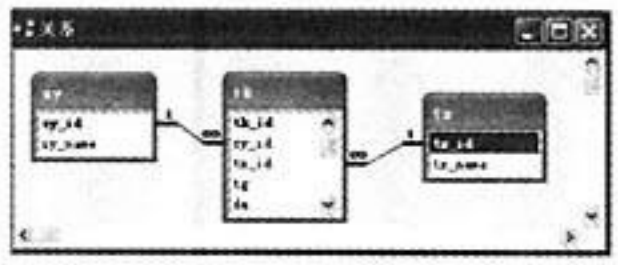


图 1 表之间的关系

2.2 数据结构

“专业技术理论考试系统”采用试卷管理类实现了试题题型、单题数据、大题数据、用户输入信息等内容的统一管理，系统结构清晰，层次分明。

2.2.1 试题类型

```
struct TQuestionType
{
    int type_id;                //题型 ID
    AnsiString type_name;  //题型名称
};
```

2.2.2 试题类型管理类

```
class TQuestionTypeManager
{
    public:
        TList * QuestionTypes;    //试题类型列表
        int    GetTypesNum()
        {
            return QuestionTypes->Count;
        };
       //加入新题型
void  AddQuestionType(AnsiString type_name,int type_id)
        {
  TQuestionType * curType = new TQuestionType();
            curType->type_id = type_id;
            curType->type_name = type_name;
            QuestionTypes->Add(curType);
        };
TQuestionTypeManager() { QuestionTypes = new TList();};
      //析构时释放题型对象实例
        ~TQuestionTypeManager()
        {
            for(int i=QuestionTypes->Count-1;i>=0;i--)
    delete (TQuestionType *)QuestionTypes->Items[i];
            delete QuestionTypes;
        };
};
```

2.2.3 试题对象类

试题对象类用来描述试卷上的每一道试题对象，包括试题的题干、答案、选项等内容：

```
//试题对象描述类
class TQuestion
{
    public:
    AnsiString    Body;              //题干内容
    TStringList*  SelectItems;     //试题选项
TStringList*  UserAnswerItems;   //用户填空题输入内容
    AnsiString    Answers;          //试题原答案
AnsiString    UserAnswers;  //用户选择、判断题输入内容
  TQuestion()
    {
       SelectItems = new TStringList();
       UserAnswerItems = new TStringList();
       UserAnswerCheck = new TStringList();
    };
    ~TQuestion()
    {
       delete SelectItems;
       delete UserAnswerCheck;
       delete UserAnswerItems;
    };
};
```

2.2.4 试题集合类

试题集合类用来管理一个大题内的所有试题，同时存储本类大题的题型和每题分值信息：

```
class TQuestionUnits  //试题集合类
{
    public:
       TQuestionType* questionType;   //题型
       float   questionScore;  //每题分值
       TList*  Questions;
       TQuestionUnits()
       {
          Questions = new TList();
       };
       ~TQuestionUnits()
       {
          for(int i=Questions->Count-1;i>=0;i--)
             delete (TQuestion *)Questions->Items[i];
          delete Questions;
       };
};
```

2.2.5 HTML 试卷管理类

HTML 试卷管理类用来实现对一次考试的全部信息进行管

理，包括设定的考试时间、用户姓名和单位、HTML 试卷题头以及试卷包含的大题等内容：

```
//HTML 试卷管理类
class THTMLPaper
{
   public:
 TList* QuestionUnits;//试题集合列表(试卷大题列表)
      int  ExamTimes;      //考试时长
      long long  ExamStartTime;//考试开始时间
 long long  ExamEndTime;      //考试结束时间
 float  UserScores;    //用户成绩
 float  PaperTotalScore;     //卷面总分
 AnsiString  UserName;     //用户名
 AnsiString  UserCompany;      //用户单位
bool   AllowMakePaper; //允许生成试卷标志
 AnsiString  PaperTitle;      //试卷标题题头
 float       SimilitudeValue; //答案相似度
                      //当相似度大于该值时得分
 //清空试卷数据
 void Clear()
 {
     UserScores = 0;
     PaperTotalScore = 0;
     for(int i=QuestionUnits->Count-1;i>=0;i--)
   delete (TQuestionUnits *)QuestionUnits->Items[i];
      QuestionUnits->Clear();
     };
     THTMLPaper()
     {
      QuestionUnits = new TList();
     };
     ~THTMLPaper()
     {
      for(int i=QuestionUnits->Count-1;i>=0;i--)
  delete (TQuestionUnits *)QuestionUnits->Items[i];
      delete QuestionUnits;
     };
};
```

2.3 系统流程

系统流程如图 2 所示。

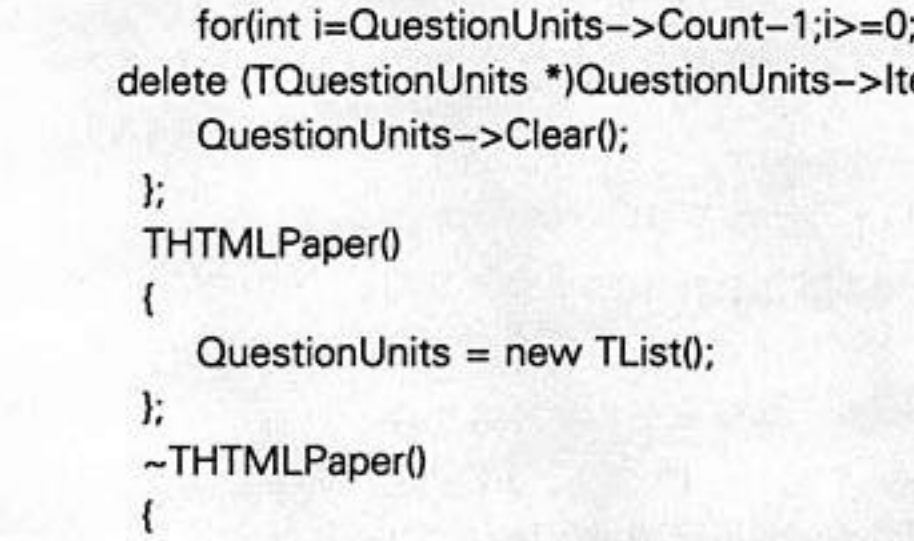


图 2 系统流程

3 关键技术编码

系统在编码实现中，重点要解决好随机抽题、HTML 试卷生成与评判、HTML 页面自动填充、系统完整性校验等关键技术。

3.1 随机抽题

“专业技术理论考试系统”数据库内有数千道试题，而每次考试，试题数目一般在 80——100 道之间，组织试卷时，就需要从试题数据库内随机的抽取要求类型、要求数目的数据记录。为此，作者设计了如下的随机数产生函数，实现所要抽取记录编号的生成。

```
//在 randRange 范围内产生 randNum 个不重复的随机数
//randRange：随机数生成范围
//randNum：要生成的随机数个数
//randArray：返回生成的随机数数列
void  MakeRands(int randRange,int randNum,int* randArray)
{
  if(randRange < randNum) return;
  int*  numArray = new int[randRange];
  for(int i=0;i<randRange;i++)
     numArray[i] = i;
  int curRange = randRange;
  for(int i=0;i<randNum;i++)
    {
      int anum = RandomRange(0,curRange);
      int tem = numArray[anum];
      numArray[anum] = numArray[curRange-1];
      numArray[curRange-1] = tem;
      curRange = curRange-1;
    }
  int index = 0;
  for(int i = randRange-1;i > randRange-1-randNum;i--)
    {
      randArray[index] = numArray[i];
      index++;
    }
  delete  []numArray;
  return;
}
```

在产生试卷的过程中，依次读取 randArray [] 数组中元素相对应的数据记录，从而快速实现试卷试题的读取。

3.2 HTML 试卷生成

系统设计生成的 HTML 试卷如图 3 所示。

专业技术理论考试（专业 1）

单位：部门 1 姓名：fsd 卷面总分：100 答题时长：45 分钟

一、填空题（共 40 题，每题 1 分，共 40 分）

1、重点防火单位和防火场所，应当制定[　　]，明确消防应急分队，定期组织有针对性的[　　]。

图 3 生成试卷效果

其 HTML 编码如下：

```
<head>
<meta  http -equiv = "Content -Type"  content = "text/html;
charset=gb2312" />
<title>temp</title>
<style type="text/css">
<! --
.STYLE1 {font-size: 24px;
      font-family: "宋体";
      font-weight: bold;}
.STYLE2 {font-size: 36px;
      font-family: "宋体";
      font-weight: bold;}
-->
</style>
<style type="text/css">
<! --
body {
      background-color: #CCFFFF;
}
-->
</style>
</head>
<body>
<p align="center" class="STYLE2">专业技术理论考试(专业 1)
</p>
<p align="center"><strong>单位：</strong>部门 1<strong>
姓名：</strong>fsd <strong>  卷面总分：</strong>100 <
strong>  答题时长</strong>：45 分钟</p>
<p align="left" class="STYLE1">一、共同科目(共 40 题,每题 1
分,共 40 分)</p>
<span  style =´mso -tab -count:1´ >     <a
name="Question_1_1"></span>
<strong>1、重点防火单位和防火场所,应当制定
<span style="mso-tab-count:1"><input type="text" size="8"
ID="100101" /></span>
,明确消防应急分队,定期组织有针对性的
<span style="mso-tab-count:1"><input type="text" size="8"
ID="100102" /></span>。<p></p>
```

将从数据库中抽出的试题按 HTML 编码格式输出到一 TstringList 对象，HTML 试卷数据就准备好了。这里特别要注意的是每个 HTML 对象（试卷中要求用户输入的部分）要进行唯一标识，以便在评卷和表单填充时可以访问到每个对象。这里采用唯一的 ID 值进行区分。ID 命名的规则为：ID 值= (大题序号+1) *100000 + (小题序号+1) *100 + (对象序号+1)。

3.3 HTML 试卷的加载

C++Builder 中无法像 C# 一样通过直接修改CppWebBrowser 的 Document 即可刷新页面显示，但可以通过流加载的方式进行刷新。流加载函数如下：

```
//从内存中刷新 HTML 文档
//CppWB: TcppWebBrowser 对象
//strList： 要刷新的页面 HTML 源码
void  TMainForm::LoadStream  (TCppWebBrowser  *CppWB,
TStringList * strList)
{
   IHTMLDocument2 *htm = NULL;
   //加载一个空的页面使得 Document 完成初始化
   CppWB->Navigate(L"about:blank");
                  if  (CppWB->Document->QueryInterface
(IID_IHTMLDocument2,(LPVOID*)&htm))
    {
      IPersistStreamInit *spPsi = NULL;
       if  (htm  ->QueryInterface  (IID_IPersistStreamInit,
(LPVOID*)&spPsi) && spPsi)
       {
                        TMemoryStream* mStream = new
TMemoryStream();
          strList->SaveToStream(mStream);
          mStream->Seek(0,0);
           TStreamAdapter  *sa  =  new  TStreamAdapter
(mStream,soReference);
          spPsi->Load(*sa);
          delete mStream;
       }
    }
}
```

3.4 HTML 试卷评判

对于用户在 HTML 表单中输入的内容，在评判时需根据对象的名称来获得该表单对象，并由得到的对象指针来访问用户输入的对象值，再将其与标准答案比较，判断正误。

下面即以表单中 ID 值为“1001001”的对象为例，说明程序中是如何获得表单对象输入值的。

```
_di_IDispatch disp;
System::DelphiInterface<IHTMLDocument2> htmldoc2;
System::DelphiInterface<IHTMLElement> htmlelem;
System::DelphiInterface<IHTMLElementCollection> htmlele
mcoll;
disp = this->WebBrowser1->Document;
htmldoc2 = disp;
htmldoc2->get_all(&htmlelemcoll);
AnsiString elementID = "1001001";   //指定对象 ID 名称
//得到指定对象 _di_IDispatch 指针
htmlelemcoll->item(TVariant(elementID),TVariant(0),&disp);
//得到 IHTMLElement 对象
htmlelem = disp;
TVariant val1;
//获得指定对象的 Value 值
htmlelem->getAttribute(WideString("Value"),0,&val1);
//获得用户输入字符串
AnsiString userinput = val1.bstrVal;
```

在系统评判过程中，就是按照顺序依次访问每个IHTMLElement对象，获取对象的Value值，同对应标准答案比对，完成试卷的评判。

3.5 HTML页面自动填充的实现

“专业技术理论考试系统”设计为单机版，当在应用该系统进行个人练习时，有时需要在练习过程中快速的查询答案，为此，设计了HTML页面自动填充功能，从而实现辅助练习者记忆知识的目的。

HTML页面的自动填充，一般可以分为按选中表单对象单个填充和所有表单对象自动填充两种方法。

（1）对选中对象的填充

```
_di_IDispatch disp;
System::DelphiInterface<IHTMLDocument2> htmldoc2;
System::DelphiInterface<IHTMLElement> htmlelem;
System::DelphiInterface<IHTMLElement> curelem;
System::DelphiInterface<IHTMLElementCollection> htmlelem
coll;
disp = this->WebBrowser1->Document;
htmldoc2 = disp;
htmldoc2->get_activeElement(&htmlelem);
BSTR tagID,tagName;
htmlelem->get_tagName(&tagName);
//判断当前选中的对象是否为可输入类型
if(AnsiString(tagName) == "INPUT")
{
    htmlelem->get_id(&tagID);
    int tagIDNum = AnsiString(tagID).ToInt();
    //根据 tagIDNum 的编码规则得到该对象相应的标准答案
    TVariant val1;
    val1 = "标准答案";
    //设定当前表单对象的值为“标准答案值”
    htmlelem->setAttribute(WideString("Value"),val1,0);
}
```

（2）对所有表单对象填充

```
//HTML 试卷中对象的 Name 值必须按规则命名
for(int i=0;i<HTMLPaper.QuestionUnits->Count;i++)
  {
    TQuestionUnits * curUnits = (TQuestionUnits *)
HTMLPaper.QuestionUnits->Items[i];
    for(int j=0;j<curUnits->Questions->Count;j++)
      {
        TQuestion * curQuestion = (TQuestion *)curUnits->
Questions->Items[j];
        //填空题所有空遍历
    for(int k=0;k<curQuestion->SelectItems->Count;k++)
          {
              AnsiString elementID = (AnsiString) ((i +1)
*100000+(j+1)*100+(k+1));//获得对象 ID
              htmlelemcoll ->item (TVariant (elementID),
TVariant(0),&disp);
              htmlelem = disp;
              TVariant val1;
              val1 = curQuestion->SelectItems->Strings[k];
//当前对象对应答案
    htmlelem->setAttribute(WideString("Value"),val1,0);
          }
      }
  }
```

4 系统运行结果

运行结果如图4-图6所示。

图4 系统登录

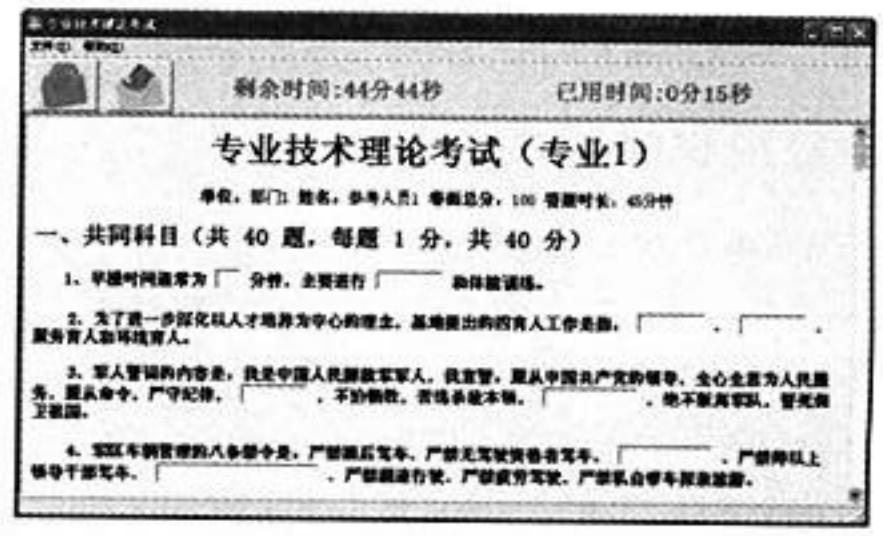


图5 HTML试卷

图6 交卷后自动阅卷结果

5 结语

从实际应用出发，详细介绍了在C++Builder环境下应用CppWebBrowser开发“专业技术理论考试系统”的过程，阐述了该类系统数据库、数据结构设计以及相关关键技术的解决方案，为基于CppWebBrowser组件开发类似系统提供了一个良好的思路。

（收稿日期：2012-10-26）

C# 开发邮件工具

李福红　于九九

摘　要：通过 C# 语言编程实现一个邮件收发器编程，进而讲解电子邮件相关协议的工作方式和原理。

关键词：C# 语言；网络编程；邮件工具

1　引言

在工作中，邮件的发送和接收应该是经常要使用到的功能的。因此知道电子邮件的应用程序的原理也是非常有必要的，在这一个专题中将介绍电子邮件应用程序的原理、电子邮件应用程序中涉及的协议和实现一个简答的电子邮件收发器程序。

2　邮件应用程序基本知识

2.1　电子邮件原理及相关协议

说到电子邮件的原理，其实和现实生活中寄邮件和寄包裹是一样的原理的。就先回顾下现实生活中寄邮件的流程吧——首先，先写好信，信封上面写好收信人的地址，写信人的地址，然后把信放到寄信箱中，邮局的人会某个时候去这个信箱中的信取出来，然后邮局的人根据信封上写的收信人地址进行转发到当地的邮局，当地邮局把信寄到收信人的信箱中（寄包裹的话可能会电话联系，像在淘宝，京东买的东西的，收货人就是通过电话联系一样），最后收信人会到自己的信箱中收取信件。上面大致是平时生活中寄信的一个流程。前面已经讲过电子邮件的原理和这个差不多的下面就介绍本专题中电子邮件的原理，大家可以和现实生活中的寄信过程进行对比，这样可以更加容易理解和掌握：

通过电子邮件应用（例如 基于客户端的 Outlook 电子邮件软件和一些基于 Web 的电子邮件系统——新浪邮箱、谷歌邮箱、QQ 邮箱等都属于电子邮件应用）将一封写好的邮件（相当于现实生活中的信，当然邮件也要写明收件人地址，邮件内容等信息的）通过电子邮件协议（SMTP，在后面的电子邮件相关协议中会介绍）发送到 SMTP 服务器（就是存储邮件的地方，相当于生活中的邮局一样），然后 SMTP 服务器根据收件人的地址通过 SMTP 协议转发到相应 SMTP 接收服务器上，(SMTP 服务器进行转发相当于现实生活中邮局的人配送信的过程，配送到收件人当地的邮局，然而现实生活中邮局都是一家，所以可以相互识别——意思就是发送到当地邮局，当地邮局会接收，并且帮助你发送到指定人的信箱中，在网上就是通过 SMTP 协议来规定这样的一个过程的，发送到别人的 SMTP 服务器上别人的服务器必须要认识发送来的邮件并接收）结束，接收端邮件服务器（POP3 服务器）把邮件存放到接受者的电子信箱内（相当于当地邮局的人把信放到收信人的邮箱中），最后收件人可以登录自己的电子信箱，再与 POP3 服务器进行连接，从 POP3 服务器上下载发送来的邮件，这样在收件人的电子信箱中就可以看到发送来的电子邮件了（这就是现实生活中收信人从自己的信箱中取信的一个过程）。

上面已经把电子邮件的原理和现实生活中寄信的过程进行对比，相信大家可以更加清楚电子邮件的原理和发送接收过程的，其实网络上的很多应用都可以以现实生活的例子去理解，这样的话可以加深对知识的理解。下面就介绍下电子邮件中的相关协议的内容：

网络上的应用的核心就是协议，因为协议让网络上的客户端相互认识发生来的数据，所以电子邮件应用也不例外，也有相关的电子邮件协议来完成发送电子邮件和接收电子邮件的过程，这些协议主要是：SMTP（简单邮件传输协议，Simple Mail Transfer Protocol）、POP3（邮局协议，Post Office Protocol）和 IMAP（网络邮件访问协议，Internet Message Access Protocol）。

（1）SMTP——SMTP 主要负责将邮件从一台机器转发至另一台机器（可以对照上面电子邮件的过程来理解 SMTP 的作用）。

（2）POP3——3 表示 POP 协议的版本，主要负责将邮件从邮箱中（POP3 服务器）传输到本地计算机。

（3）IMAP——现在常用的版本为第四版本，即 IMAP4，主要负责邮件的检索和处理功能，客户端不需要下载邮件到本地计算机，可直接从邮件客户端软件对服务器上的信件和文件目录进行操作，它是 POP3 的替代协议。

2.2　邮件系统分类

邮件系统主要分为两类的——基于客户端的邮件系统和基于 Web 浏览器的邮件系统。Office OutLook 就是基于客户端的邮件客户端系统，而像我们经常使用的 QQ 邮箱、新浪、网易邮箱等都是属于基于 Web 浏览器的邮件系统，基于客户端的邮件系统的收发过程，通过如图 1 所示来描述。

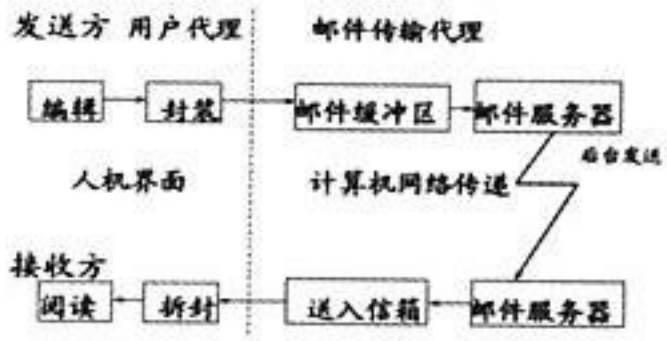


图 1　基于客户端的邮件收发过程

发送方通过邮件客户端，将编辑好的邮件向邮件服务(SMTP 服务器，在发送过程中也叫发送端邮件服务器）发送，发送端邮件服务器根据收件人的地址来识别接收端邮件服务器(POP3 服务器)，然后向 POP3 服务器发送邮件信息，接收端邮件服务器将邮件存放在接收者的电子信箱中，并告知接收者有新邮件，接收者通过邮件客户端与 POP3 服务器连接后，就可以查看新邮件。

然而，基于 Web 浏览器的邮件系统与基于客户端的邮件系统不同的地方如下：

(1) 基于 Web 浏览器邮件系统用户代理 (代理的概念也就是用户不是直接与服务器进行通信，而是通过代理的方式，让代理去与服务器通信，然后用户在从代理中获的服务器的信息，代理也就是中间人的作用，相当于生活中中介，在.net 中很多技术都用到了代理，例如委托的概念其实也就是代理的一个概念的) 是 Web 浏览器，基于客户端的邮件系统而是邮件客户端应用程序，一般是 Windows Form 程序。

(2) 浏览器发送邮件到 SMTP 服务器和从 POP3 服务器中获得邮件的方式都是通过 HTTP 协议来实现，与基于客户端的邮件系统不同（基于客户端的邮件系统发送通过 SMTP 协议或 ESMTP（Extended SMTP)，获得通过 POP3 或 IMAP 协议)。

2.3　目前主要的电子邮件服务系统

电子邮件服务系统——就是向大家提供邮箱服务的服务系统，这样的系统当然是由专门的公司进行研发的，一般叫这样的公司为邮件服务商，平常使用的网易邮箱，新浪、Gmail 邮箱等都是建立在电子邮件服务系统。现在主要电子邮件服务系统主要有下面几种：

(1) 基于 Postfix/Qmail 的邮件系统。例如，雅虎邮箱基于 Qmail 系统。

(2) 微软 Exchange 邮件系统。

(3) IBM Lotus Domino 邮件系统。

(4) Scalix 邮件系统。

(5) Zimbra 邮件系统。

(6) MDeamon 邮件系统。

3　.Net 平台对邮件发送功能的支持

在.Net 类库中，在 System.Net.Mail 命名空间下定义了对邮件处理的类，这样使邮件的发送更加方便（这些类也就是对 SMTP 协议的封装，使我们更好地区编程，只需要使用类中的方法和属性等去完成邮件的发送，避免写复杂的 SMTP 协议的命令)。

C# 提供了一系列邮件相关的类，在这里就不一一介绍了，大家可以参考 MSDN 去看每个类的使用，并且在后面程序的实现部分也会有详细的注释去介绍程序中使用到类的使用。需要注意的是：C# 提供的类中只有 SMTP 的字样，却没有 POP3 这样的字样，这说明.Net 类库本身中并没有提供对 POP3 协议的封装类，但是可以使用 Jmail 组件来完成从 POP3 服务器中收取邮件的功能，具体的使用将在后面的邮件收发器程序中邮件的接收部分介绍。

4　邮件收发器程序的实现

4.1　邮件发送功能

4.1.1 SMTP 协议

SMTP 协议是用于电子邮件的传输的协议，电子邮件是通过 SMTP 服务器进行发送的，SMTP 服务器的默认端口为 25，通常发送邮件有两种方式———一种是不使用客户端认证，即客户端可以使用匿名发送邮件（这种方式叫做 SMTP)；另一种是客户端必须提供用户名和密码认证（这种方式叫做 ESMTP, Extended SMTP）目前大部分邮件服务器采用用户名和密码认证的方式。

客户端发送邮件过程为——先通过客户端软件（本程序中的邮件收发器）将邮件发送到 SMTP 服务器，然后再由 SMTP 服务器发送到目标 SMTP 服务器。下面介绍 SMTP 协议的内容：

SMTP 协议总共定义了 14 个命令，命令由命令码和气候的参数域组成，不区别大小写（通过前面专题的讲述可以得出各个协议的命令组成都差不多的)，表 1 就简单介绍下 5 个常用的命令码：

表 1　SMTP 协议常用命令

名称	解释
HELO 或 EHLO	发送连接到服务器的命令，EHLO 主要用于与 ESMTP 服务器建立连接时发送的命令
MAIL FROM	指定发件人的邮件地址
Rcpt to	指定收件人的邮件地址
Data	指定邮件正文内容，邮件内容以单独一行？"." 表示接触
Quit	关闭与服务器的连接，然后退出

电子邮件由信封、首部、正文和结束符号 4 部分组成，下面就具体介绍下这 4 个部分的内容：

(1) 信封

信封包括发信人的邮件地址和接收人的邮件地址，具体对应两条 SMTP 命令——Mail from: mytest1989@sina.cn (发信人的地址) 和 Rcpt to: test@126.com。

(2) 首部

首部中常用的命令有：

1) Subject：<邮件主题>——表示邮件的主题。

2) Date:<时间>——表示发邮件的时间。

3) reply-to:<邮件地址>——表示邮件的回复地址。

4) Content-Type:<邮件类型>——表示邮件包含文本、HTML 超文本和附件的类型。

5) X-Priority:<邮件优先级>——表示邮件发送的优先级，优先级为 3 表示为普通邮件；如 X-Priority：3。

(3) 正文

正文当然指的就是邮件的内容了，用 Data 命令指定，首部以一个空行结束，下面就是正文部分。

(4) 结束符号

邮件以 "." 结束，

接收方收到 SMTP 命令之后，会给出一个响应码，每个命令都只有一个响应码，SMTP 响应码也是由 3 位数字组成，后面附加一些文本信息，响应信息的格式为：

响应码<空格>文本信息<回车换行>

客户端发出一条命令后，服务器端返回一个响应，发送者在发送下一条命令前必须等待服务器的响应，成功接收到响应码后才继续发送命令。

附：SMTP 常用的响应码：

响应码　解释 响应码　解释

211　系统状态或系统帮助响应　421　服务未就绪，关闭了传输通道

214　帮助信息　501　参数格式错误

220　服务就绪　502　命令不可实现

221　服务关闭传输通道　535　用户验证失败

235　用户验证成功　553　邮箱名不可用，要求的操作未执行

334　等待用户输入验证　554　操作失败

354　开始邮件输入

4.1.2 邮件的发送过程

(1) 客户端与服务器建立连接（该步中客户端首先发送 EHLO local 连接命令，服务器如果返回 "220" 响应码表示服务器准备就绪了，客户端再继续发送 "Auto login" 命令，请求登录，服务器收到命令后返回 "334" 响应码，表示要输入用户名，之后客户端发送用户名命令，等到响应后再发送密码命令，具体在程序的实现中也会有注释。)

(2) 客户端发送邮件的信封。

(3) 开始发送邮件数据，(包括邮件首部，正文和结束符号，注：结束符号要单独占一行，表示邮件发送结束)。

(4) 客户端与服务器断开连接。

4.1.3 发送功能的实现代码

相信有了上面的理论解释邮件发送的过程后，实现邮件发送的功能并不难的，并且.net 类库中 SMTPClient 类封装了 SMTP 协议，使得实现邮件发送功能就不要记住那些具体的命令了，只需要使用该类中提供的方法来完成邮件的发送（当然你也可以通过发送命令的方式实现，SMTPClient 类的方法也是完成发送命令功能而已的），下面是邮件发送功能的核心代码：

```
#region 邮件发送功能代码
        // 添加附件
private void btnAddFile_Click(object sender, EventArgs e)
        {
OpenFileDialog openFileDialog = new OpenFileDialog();
            openFileDialog.CheckFileExists = true;
            // 只接受有效的文件名
            openFileDialog.ValidateNames = true;
            // 允许一次选择多个文件作为附件
            openFileDialog.Multiselect = true;
            openFileDialog.Filter = "所有文件(*.*)|*.**";
if (openFileDialog.ShowDialog() ! = DialogResult.OK)
            {
                return;
            }
            if (openFileDialog.FileNames.Length > 0)
            {
                // 因为这里允许选择多个文件,所以这里用
//AddRange 而没有用 Add 方法
cmbAttachment.Items.AddRange(openFileDialog.FileNames);
            }
        }
        // 删除附件
private void btnDeleteFile_Click(object sender, EventArgs e)
        {
            int index = cmbAttachment.SelectedIndex;
            if (index == -1)
            {
    MessageBox.Show (" 请选择要删除的附件！ ", " 提示",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                return;
            }
            else
            {
                cmbAttachment.Items.RemoveAt(index);
            }
        }
        // 发送邮件
private void btnSend_Click(object sender, EventArgs e)
        {
            this.Cursor = Cursors.WaitCursor;
            // 实例化一个发送的邮件
             // 相当于与现实生活中先写信,程序中把信(邮
//件)抽象为邮件类了
    MailMessage mailMessage = new MailMessage();
            // 指明邮件发送的地址,主题,内容等信息
```

```
        // 发信人的地址为登录收发器的地址,这个收发器相当
//于我们平时 Web 版的邮箱或者是 OutLook 中配置的邮箱
  mailMessage.From = new MailAddress(tbxUserMail.Text);
            mailMessage.To.Add(txbSendTo.Text);
            mailMessage.Subject = txbSubject.Text;
      mailMessage.SubjectEncoding = Encoding.Default;
            mailMessage.Body = richtbxBody.Text;
    mailMessage.BodyEncoding = Encoding.Default;
            // 设置邮件正文不是 Html 格式的内容
            mailMessage.IsBodyHtml = false;
            // 设置邮件的优先级为普通优先级
            mailMessage.Priority = MailPriority.Normal;
               //mailMessage.ReplyTo = new MailAddress
(tbxUserMail.Text);
            // 封装发送的附件
    System.Net.Mail.Attachment attachment = null;
            if (cmbAttachment.Items.Count > 0)
            {
     for (int i = 0; i < cmbAttachment.Items.Count; i++)
                {
  string fileNamePath = cmbAttachment.Items[i].ToString();
                    string extName = Path.GetExtension
(fileNamePath).ToLower();
       if (extName == ".rar" || extName == ".zip")
                    {
          attachment = new System.Net.Mail.Attachment
(fileNamePath, MediaTypeNames.Application.Zip);
                    }
                    else
                    {
        attachment = new System.Net.Mail.Attachment
(fileNamePath,MediaTypeNames.Application.Octet);
                    }
      // 表示 MIMEContent-Disposition 标头信息
// 对于 ContentDisposition 具体类的解释大家可以参考 MSDN
                    // 这里就不重复贴出来了,给个地址: http:
//msdn.microsoft.com/zh-cn/library/System.Net.Mime.Conte
ntDisposition.aspx (着重看备注部分)
   ContentDisposition cd = attachment.ContentDisposition;
     cd.CreationDate = File.GetCreationTime(fileNamePath);
cd.ModificationDate = File.GetLastWriteTime(fileNamePath);
  cd.ReadDate = File.GetLastAccessTime(fileNamePath);
                // 把附件对象加入到邮件附件集合中
    mailMessage.Attachments.Add(attachment);
                }
            }
            // 发送写好的邮件
            try
            {
// SmtpClient 类用于将邮件发送到 SMTP 服务器
            // 该类封装了 SMTP 协议的实现,
            // 通过该类可以简化发送邮件的过程,只需要
//调用该类的 Send 方法就可以发送邮件到 SMTP 服务器了。
                smtpClient.Send(mailMessage);
                MessageBox.Show (" 邮件发送成功! ", "提
示", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
            catch(SmtpException smtpError)
            {
                MessageBox.Show (" 邮件发送失败:[" +
smtpError.StatusCode + "];["
  + smtpError.Message+"];\r\n["+smtpError.StackTrace+"]."
                    ,"错误",MessageBoxButtons.RetryCancel,
MessageBoxIcon.Error);
            }
            finally
            {
                mailMessage.Dispose();
                this.Cursor = Cursors.Default;
            }
        }
        #endregion
```

4.2 邮件接收功能

4.2.1 POP3 协议

前面介绍了邮件的发送，当然接收者需要登录邮箱来查看收到的邮件了，此时就必有有一个协议去读取服务器上邮件，POP3 就是这样的一个协议。还有两外一种协议也是用来接收邮件的——IMAP 协议，它与 POP3 协议区别有：

（1）IMAP 使用的端口号是 143 而 POP3 邮件服务器通过监听 110 端口来提供 POP3 服务。

（2）IMAP 允许客户端在邮件服务器上建立文件夹来保持邮件，而不用把邮件下载到本地。而 POP3 需要把邮件下载到本地。和 SMTP 协议一样，客户端要通过 POP3 协议从 POP3 服务器上获取邮件，也需要先与 POP3 服务器建立 TCP 连接，等待服务器向客户端发送确认信息表明连接成功时，客户端才可以继续发送命令给服务器来获取邮件，在 POP3 协议中，规定的命令也是几十条的，每条命令由命令和参数两部分组成，都是以回车换行结束，并且命令和参数之间由空格分隔，命令通常也是由 3–4 个字母组成，参数最多可以为 40 个字符的长度，而服务器的响应信息是由一个状态码和可能附加信息的字符组成，所有的响应信息也是以回车换行结束的。状态码和其他协议定义的状态码有点不一样，POP3 服务器响应的状态码有两种——“+OK” (确定) 和" -ERR" （失败）。这样客户端可以通过检查响应的状态码所包含的字符来判断服务器是否响应客户端发送的命令，即响应信息中包含 “+OK” 表示成功响应，包含 “-ERR” 表示服务器未响应。同时在程序的实现中大家可以通过 Debug 来查看响应消息的组成，这样可以加深理解。

4.2.2 邮件接收的过程

客户端从服务器接收邮件的过程主要经历 3 个状态：授权

状态、操作状态和更新状态

(1) 授权状态——客户端发送与 POP3 服务器的 TCP 连接请求，服务器接收后发送一个响应确认信息之后，此时客户端需要发送正确的用户名和密码进行确认，因为在邮件服务器上有很多用户邮箱，只有提供正确的用户名和密码才有权限访问自己的邮箱，就像现实生活中邮箱的钥匙一样的。

发送用户名命令： USER test@s126.com

发送密码命令： PASS ****** (这两个命令都在代码中有给出的，大家可以参考代码来理解邮件的接收过程)

(2) 操作状态——如果客户端提供了正确的用户名和密码，则授权状态也就通过了，就相当于打开了在服务器上自己的邮箱，现在用户就有权限进去下载，查看和删除邮件等操作，然后在现实生活中的取邮件和删除邮件都很简单（只要打开了邮箱门，用手去拿就可以了），然后在网络应用上，这些操作都需要发送 POP3 命令给服务器，服务器接收到命令后再给出响应。操作中常用的命令有：

1) STAT 命令——该命令从服务器中获取邮件总数和总字节数，服务器响应命令返回邮件总数和总字节数

如：

?

客户端发送 POP3 命令： STAT

服务器响应命令： +OK 2 1340
服务器响应命令。

2) LIST 命令——该命令从服务器中获得邮件列表和大小，服务器响应命令返回列出邮件列表和大小。

如：

?

客户端发送 POP3 命令：LIST

服务器响应命令： +OK 2 message (1430 octect)

服务器响应命令：1 700

服务器响应命令：2 730

服务器响应命令：<一个空行>

3) RETR 命令—— 该命令从服务器中获得一个邮件，格式为 RETR <邮件编号>

如：

?

客户端发送 POP3 命令：RETR 1

服务器响应命令： 700 octets

服务器响应命令：<邮件头和内容>

服务器响应命令： <空行>

4) DELETE 命令——该命令告诉服务器将邮件标记为删除。（此时只是逻辑删除）

(3) 更新状态——客户端发送 QUIT 命令后，此时就进入更新状态，POP3 服务器释放在操作状态中取得的资源，并将逻辑删除的邮件进行物理删除，然后关闭与客户端的 TCP 连接。这样整个邮件处理的过程就结束了。

4.2.3 接收功能的实现代码

有了前面接收邮件过程的介绍，再参考代码的实现，相信大家可以更好的理解客户端从 POP3 服务器中获取邮件的过程，由于.net 类库并没有封装 POP3 协议的实现类，要实现邮件的获取可以采用发送命令的方式，也可以使用 Jmail 组件，这个组件其实就是 POP3 协议的封装类，既然微软没有做，其他公司做好后来并简单的实现邮件的接收的一个类库罢了。然后在使用这个组件的过程中出现了好几个问题，在源码中都做了解释，大家可以下载源代码后查看。

实现邮件接收的核心代码如下：

```
// 登录邮箱(这里是本程序——邮件收发器)
private void btnLogin_Click_1(object sender, EventArgs e)
        {
            // 与 POP3 服务器建立 TCP 连接
            // 建立连接后把服务器上的邮件下载到本地
            // 设置当前界面的光标为等待光标(就是看到的
//一个动的圆形)
            Cursor.Current = Cursors.WaitCursor;
            lsttbxStatus.Items.Clear();
            try
            {
// POP3 服务器通过监听 TCP110 端口来提供 POP3 服务的
                // 向 POP3 服务器发出 tcp 请求
    tcpClient = new TcpClient(tbxPOP3Server.Text, 110);
                lsttbxStatus.Items.Add("正在连接...");
            }
            catch
            {
                MessageBox.Show ("连接失败", "错误",
MessageBoxButtons.RetryCancel, MessageBoxIcon.Error);
                lsttbxStatus.Items.Add("连接失败！");
                return;
            }
            // 连接成功的情况
            networkStream = tcpClient.GetStream();
                        streamReader = new StreamReader
(networkStream, Encoding.Default);
                        streamWriter = new StreamWriter
(networkStream, Encoding.Default);
            streamWriter.AutoFlush = true;
            string str;
            // 读取服务器返回的响应连接信息
            str = GetResponse();
            if (CheckResponse(str) == false)
            {
    lsttbxStatus.Items.Add("服务器拒接了连接请求");
                return;
            }
            // 如果服务器接收请求
```

```
            // 向服务器发送凭证——用户名和密码
            // 向服务器发送用户名,请求确认
            lsttbxStatus.Items.Add("核实用户名阶段...");
            SendToServer("USER " + tbxUserMail.Text);
            str = GetResponse();
            if (CheckResponse(str) == false)
            {
                lsttbxStatus.Items.Add("用户名错误.");
                return;
            }
            // 用户名审核通过后再发送密码等待确认
            // 向服务器发送密码,请求确认
            SendToServer("PASS "+txbPassword.Text);
            str = GetResponse();
            if (CheckResponse(str) == false)
            {
                lsttbxStatus.Items.Add("密码错误！");
                return;
            }
        lsttbxStatus.Items.Add("身份验证成功,可以开始会话");
    // 向服务器发送 LIST 命令,请求获得邮件列表和大小
            lsttbxStatus.Items.Add("获取邮件....");
            SendToServer("LIST");
            str = GetResponse();
            if (CheckResponse(str) == false)
            {
                lsttbxStatus.Items.Add("获取邮件列表失败");
                return;
            }
            lsttbxStatus.Items.Add("邮件获取成功");
            // 窗口控件控制
            tabControlMyMailbox.Enabled = true;
            btnReadMail.Enabled = false;
            btnDownLoad.Enabled = false;
            btnDeleteMail.Enabled = false;
             // 登录成功后实例化邮件发送对象,以便后面完
//成发送邮件的操作
            // 实例化邮件发送类(SmtpClient)对象
            if (smtpClient == null)
            {
                smtpClient = new SmtpClient();
                smtpClient.Host = tbxSmtpServer.Text;
                smtpClient.Port = 25;
                // 不使用默认凭证,即需要认证登陆
                smtpClient.UseDefaultCredentials = false;
        smtpClient.Credentials = new NetworkCredential
(tbxUserMail.Text, txbPassword.Text);
smtpClient.DeliveryMethod = SmtpDeliveryMethod.Network;
            }
            // 登录成功后,自动接收新邮件
            // 开始接收邮件
            try
            {
                btnRefreshMailList.PerformClick();
            }
            catch
            {
        MessageBox.Show (" 读取邮件列表失败！ ", " 错误",
MessageBoxButtons.RetryCancel, MessageBoxIcon.Error);
            }
            lsttbxStatus.Items.Add("登录成功！ ");
        lsttbxStatus.TopIndex = lsttbxStatus.Items.Count - 1;
            Cursor.Current = Cursors.Default;
            // 窗口控件控制
            richtbxMailContentReview.Enabled = true;
            tbxUserMail.Enabled = false;
            txbPassword.Enabled = false;
            btnLogin.Enabled = false;
            btnLogout.Enabled = true;
            tbxSmtpServer.Enabled = false;
            tbxPOP3Server.Enabled = false;
            btnReadMail.Enabled = true;
            btnDownLoad.Enabled = true;
            btnDeleteMail.Enabled = true;
            tabControlMyMailbox.Focus();
        }
        #region 处理与 POP3 服务器交互事件
        // 获取服务器响应的信息
        private string GetResponse()
        {
            string str = null;
            try
            {
                str = streamReader.ReadLine();
                if (str == null)
                {
    lsttbxStatus.Items.Add("连接失败——服务器没有响应");
                }
                else
                {
        lsttbxStatus.Items.Add("收到:[" + str + "]");
                    if (str.StartsWith("-ERR"))
                    {
                        str = null;
                    }
                }
            }
            catch(Exception err)
            {
    lsttbxStatus.Items.Add("连接失败:[" + err.Message + "]");
            }
            return str;
        }
        // 检查响应信息
```

```
private bool CheckResponse(string responseString)
{
    if (responseString == null)
    {
        return false;
    }
    else
    {
        if (responseString.StartsWith("+OK"))
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
// 向服务器发送命令
private bool SendToServer(string str)
{
    try
    {
        // 这里必须使用 WriteLine 方法的,因为 POP3
//协议中定义的命令是以回车换行结束的
        // 如果客户端发送的命令没有以回车换行结
//束,POP3 服务器就不能识别,也就不能响应客户端的请求了
        // 如果想用 Write 方法,则 str 输入的参数字符
//中必须包含"\r\n",也就是回车换行字符串。
        streamWriter.WriteLine(str);
        streamWriter.Flush();
        lsttbxStatus.Items.Add("发送:[" + str + "]");
        return true;
    }
    catch(Exception ex)
    {
lsttbxStatus.Items.Add("发送失败:[" + ex.Message + "]");
        return false;
    }
}
#endregion
```

4.3 程序运行结果

首先输入邮箱名和密码登录到 POP3 服务器来获取邮件列表的演示，如图 2 所示。

然后在邮件列表中选中一个邮件进行阅读，然后进行回复邮件的操作演示（邮件的发送都可以附加附件发送出去）；阅读邮件的界面，如图 3 所示。

回复邮件的界面，如图 4 所示。

同时点击发送按钮后，就可以把邮件发送到 sina 的 SMTP 服务器上，再由新浪的 SMTP 服务器转发到 QQ 的 SMTP 服务器，QQ 的 POP3 服务器中 QQ 的 SMTP 服务器获取收到的邮件，当 QQ 用户输入正确的邮箱名和密码后就可以从 QQ 的 POP3 服务器上获取收到的邮件。

点击发送按钮后成功发送邮件的图片，如图 5 所示。

图 2 获取邮件列表

图 3 阅读邮件

图 4 邮件回复

图 5 邮件发送

5 结语

通过介绍邮件发送和接收的原理，希望能熟悉简单邮件收发器的功能。

（收稿日期：2012-11-15）

ASP.NET 开发数据库三层架构系统初探

刘 林

摘 要：通过一个新闻发布系统的设计与实现并给出部分代码实例，对基于数据库的三层架构设计进行了初探，并针对设计过程中出现的若干问题进行了分析与解决。

关键词：ASP.NET 技术；C# 语言；数据库；三层架构；VS2008 工具

1 引言

经过国家中职师资培训项目中的网站建设与管理专业的学习，要求最后设计与开发一个基于数据库的新闻发布系统。在开发的过程中，了解三层架构与二层架构的区别，特别是三层架构在后期维护上的的巨大优势。现就 VS2008 下 ASP.NET+C# 环境中三层架构的设计，以及在数据库设计和代码编写等方面得到的经验与教训进行初步探讨。

2 二层与三层

要使用三层模式进行开发，首先要明白其与二层的区别：

（1）二层架构是传统的用户界面（UI）直接访问数据库服务器的模式，以本项目开发环境为例，虽然 ASP.NET 本身已经进行了设计界面（.aspx 文件）与代码页（.cs 文件）的分离，但涉及到数据库，对数据库的连接、操作的定义和实现等，都在同一个.cs 代码页中，增加了程序的复杂性，降低了可维护性。

（2）所谓“三层架构”，即在原来二层架构的用户界面（UI）和数据库服务器之间，人为地加入中间层（即组件层），而中间层又可以划分为业务逻辑层（BLL）、数据访问层（DAL）和数据对象模型层（Model），其中的数据对象模型层（Model）可以把表当做一个对象来处理，充分体现了面向对象的思想。这样，就能使代码部分有效地分层，每层各司其职，降低了程序的复杂性，减少了系统维护的开销和难度。如图 1 所示。

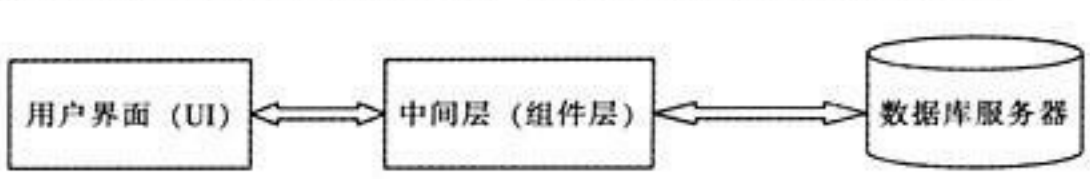


图 1 三层架构模型

（3）三层架构模式因为“高内聚、低耦合”的特性，使其在代码的复用性、灵活性和可维护性上明显优于二层模式，而且由于分层合理，便于多个开发人员之间的协作，每个人只需关注自己负责的那层就行。例如：网站前台设计人员只需遵循事先制定的接口标准，无需了解关于数据库访问的具体技术细节，就可以同数据库等设计人员实现并行开发。

3 系统功能与结构

根据需求，新闻发布系统由前台访问者和后台管理员两大部分组成，前台访问者具有的基本功能：浏览新闻、按照标题或新闻内容在本站内搜索新闻、按照已建好类别进行新闻的分类显示、对某条新闻进行评论；后台管理者具有的基本功能：注册与登录后台管理系统、对新闻的管理（添加、更新、删除）、评论的管理、新闻类别的管理和用户的管理。系统功能结构如图 2 所示。

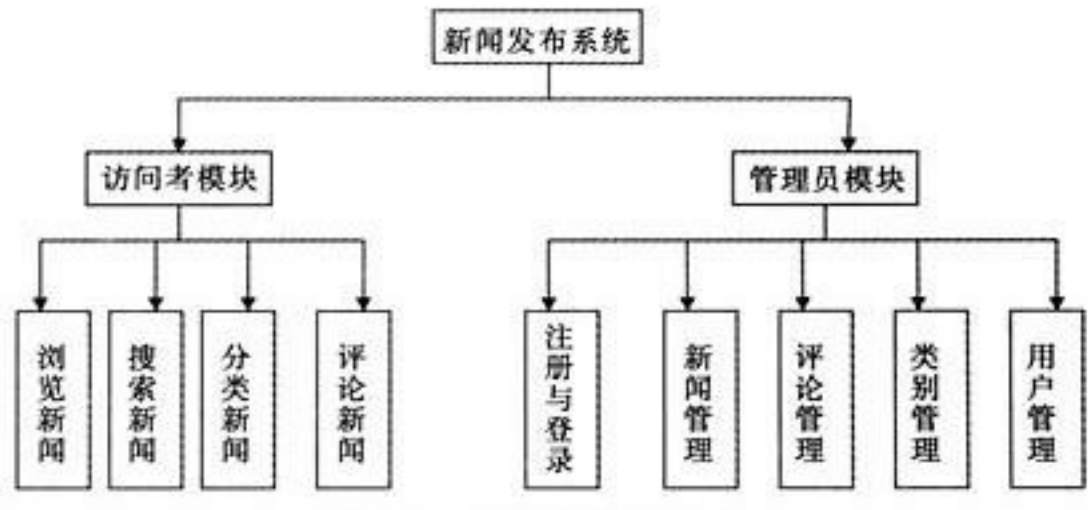


图 2 系统功能结构图

4 数据库

本系统采用的数据库系统为 SQL Server 2005，根据功能分析，建立数据库“newsPub”，由新闻表（news）、新闻分类表（category）、评论表（comment）以及用户信息表（users）构成。如表 1~表 4 所示。

表 1 新闻表（news）结构

列名	数据类型	备注	允许为 null
id	int（自动编号）	新闻编号、主键	No
title	varchar（100）	新闻标题	No
content	image	新闻内容，可存非文本	Yes
createTime	datetime	上传时间	No
caId	int	新闻分类	No
uploader	varchar（50）	上传者	No

表 2 分类表（category）结构

列名	数据类型	备注	允许为 null
id	int（自动编号）	类别编号、主键	No
name	varchar（50）	新闻类别名	No

表 3 评论表（comment）结构

列名	数据类型	备注	允许为 null
id	int（自动编号）	评论编号、主键	No
content	image	评论内容	No
createTime	datetime	评论时间	No
ip	varchar（15）	评论者 ip	No
newsId	int	被评论新闻编号	No

表 4 用户表（users）结构

列名	数据类型	备注	允许为 null
id	int（自动编号）	用户编号、主键	No
name	varchar（20）	注册用户名	No
pwd	varchar（40）	密码	No
email	varchar（50）	用户邮箱	No
uAuthority	int	用户权限	No

5 中间层

5.1 新建空白解决方案

打开 Visual Studio 2008，选择菜单“文件-->新建-->项目-->其他项目类型-->Visual Studio 解决方案-->空白解决方案”，命名为“三层架构新闻发布系统”。

5.2 创建 Model 层（实体层）

（1）添加类库。

在“三层架构新闻发布系统”解决方案上执行“右击-->添加-->新建项目-->Visual C#-->类库”，命名为“Model”。

（2）关于类库的说明。

在 Model 层的设计中，每一个类对应数据库中的一张表，而类中的每个属性对应表中的一个字段（名字不必相同，但要有对应关系）。在“newsPub”数据库中共设计了 4 个表，相应地定义了 4 个实体类 news.cs、category.cs、comment.cs 和 users.cs。

（3）定义实体类（以“News”类为例，其他 3 个实体类的定义类似）。

将生成的“class1.cs”文件重命名为“News.cs”，打开“News.cs”，代码如下：

```
namespace Model
{
public class News
{
   //新闻的 id 字段
  private string id;
  //设置新闻的 id 属性
   public string Id
   {
      get { return id; }
      set { id = value; }
   }
       //新闻的标题字段
   private string title;
  //设置新闻的标题属性
   public string Title
   {
      get { return title; }
      set { title = value; }
   }
  //其他字段与属性定义方法相同(略)
  //空的构造函数
   public News() { }
       //带参数的构造函数 1
    public News (string title, string content, string caid,
string uploader)
   {
      //标题等字段的初始化
   this.title = title;
      this.content = content;
      this.caId = caid;
      this.uploader = uploader;
   }
  //带参数的构造函数 2
    public News(string id, string title, string content, string
caid, string uploader)
   {
      this.id = id;
      this.title = title;
      this.content = content;
      this.caId = caid;
      this.uploader = uploader;
   }
}
```

5.3 创建 DAL 层（数据访问层）

5.3.1 本层作用

该层能够直接操作数据库，包括查找数据、增加记录、更新记录、删除记录等，对应 Model 类库中的每个表类，都有一个相应的 DAL 层访问类。DAL 层能够对它的上一层 BLL（业务逻辑层）提供访问数据库的接口，而又对 BLL 层屏蔽了数据库操作的具体实现。

5.3.2 具体实现

（1）添加“DAL”类库，方法与“Model”类库添加方法相同。

（2）定义“DAL”类库中的类，本系统包括以下5个类：

1）SQLHelper.cs (SQL数据库助手类)。

这个类实现了数据库连接的建立、打开连接、执行参数各异的SQL语句，同时为本层中的其他类提供了参数传递接口，其他类只需提供各种查询所需的参数，就可完成查询，从而进一步提高了公共代码的复用程度。代码如下：

```
//命名空间的引用
using System.Data;
using System.Data.SqlClient;
namespace DAL
{
    public class SQLHelper
    {
        private SqlConnection conn=null;
        private SqlCommand cmd=null;
        private SqlDataReader sdr=null;
        //通过构造函数建立连接
        public SQLHelper()
        {
            //连接字符串根据机器具体情况进行配置
        string strConn= @"server=.server;database=newsPub;
uid=sa;pwd=123456";
            conn = new SqlConnection(strConn);
        }
        //打开连接
        private SqlConnection OpenConn()
        {
            //如果连接关闭,则打开连接
      //ConnectionState 包含在 System.Data 中
       if (conn.State == ConnectionState.Closed)
            {
                conn.Open();
            }
            return conn;
        }
        //执行不带参数的 SQL 语句或存储过程并关闭连接
public int ExecuteNonQuery(string cmdText, CommandType ct)
    {
        int result;
        try
        {
            cmd = new SqlCommand(cmdText, OpenConn());
            cmd.CommandType = ct;
            res = cmd.ExecuteNonQuery();
        }
        catch (Exception ex)
        {
            throw ex;
        }
        finally
        {
            if (conn.State == ConnectionState.Open)
            {
                conn.Close();
            }
            return result;
        }
    //其他数据库操作方法略
    }
}
```

2）NewsDAO.cs (新闻表的数据访问对象类)，代码如下：

```
//命名空间的引用
using System.Data;
using System.Data.SqlClient;
//引用 Model 之前,先要在项目中添加对 Model 的引用
using Model;
namespace DAL
{
    //定义新闻数据访问对象类
      public class NewsDAO
    {
        //通过引用 Model 类库,能够定义 SQLHelper 类型的对
//象 sqlhelper
        private SQLHelper sqlhelper;
        //在构造函数中创建数据库助手类对象,建立连接
        public NewsDAO()
        {
       sqlhelper = new SQLHelper();
        }
        //选择全部新闻方法的定义
        public DataTable SelectAll()
        {
            DataTable dt = new DataTable();
       //查询 news 表中的所有新闻,并按上传时间降序排列
string sql = "select * from news order by createTime desc";
                    //调用 SQLHelper 类中的执行查询方法,并传
//递给具体的参数
dt = new SQLHelper().ExecuteQuery(sql, CommandType.Text);
                    //返回查询结果,类型是 DataTable 类表对象
            return dt;
        }
//根据新闻 ID 取出新闻内容的方法,调用 SQL 存储过程查询
        public News SelectById(string id)
        {
            //构造 News 类的对象实例 n,作为返回值类型,对应
//数据表中的一条记录
        News n = new News();
        //构造 DataTable 类的对象实例 dt,存储数据库的查询结果
```

```
        DataTable dt = new DataTable();
//预先在 SQL Server 2005 中定义了存储过程"news_selectBy
//Id",并把名字赋给//查询参数 cmdText
        string cmdText = "news_selectById";
       //将参数中的字符串类型的 id 转换为数据库中定义的 id
//类型:int
        int idsql = Convert.ToInt32(id);
        SqlParameter[] paras = new SqlParameter[]{
           new SqlParameter("@id",idsql)
        };
        //传递参数,调用 sqlhelpper 对象执行查询
      dt = sqlhelper.ExecuteQuery (cmdText, paras,
CommandType.StoredProcedure);
    //判断记录中的 content 字段值是否为空
        if (dt.Rows[0]["content"] == DBNull.Value)
        {
           n.Content = "内容暂为空!";
        }
        else
        {
             n.Content = System.Text.Encoding.Default.
GetString(dt.Rows[0]["content"] as byte[]);
         }
        n.Id = id;
        n.Title = dt.Rows[0]["title"].ToString();
        n.CreateTime = dt.Rows[0]["createTime"].ToString();
        n.CaId = dt.Rows[0]["caId"].ToString();
        n.Uploader = dt.Rows[0]["uploader"].ToString();
        return n;
      }
//其他 news 表的操作略
    }
}
```

3）CategoryDAO.cs (类别表的数据访问对象类)。

4）CommentDAO.cs (评论表的数据访问对象类)。

5）UsersDAO.cs (用户表的数据访问对象类)。

5.4 创建 BLL 层 (业务逻辑层)

5.4.1 本层作用

在 DAL 层的支持下，为系统的每个功能模块设计一个类，以实现此模块的业务逻辑，BLL 层是 UI 和 DAL 层之间的桥梁，负责数据参数的传递和处理工作。因为有了底层 DAL 的支持，BLL 层并不涉及具体的数据库访问操作，使逻辑功能更清晰，利于逻辑功能的增删改等变化。

5.4.2 具体实现

(1) 添加 "BLL" 类库，方法与 "Model" 类库添加方法相同。

(2) 定义 "BLL" 类库中的类，本系统包括以下 4 个类：

1）NewsManage.cs (新闻管理业务逻辑类)，代码如下：

```
//命名空间的引用
using System.Data;
//DAL 的引用在整个 BLL 类库中都是必须的,BLL 层通过 DAL
//层中的类的对象实例实现了对 DAL 中方法的调用,同样引用
//前先要在项目中添加对 DAL 的引用。
using DAL;
//Model 的引用根据需要,并不是必须的
using Model;
namespace BLL
{
    public class NewsManage
    {
        private NewsDAO ndao = null;
        //建立与 DAL 层的数据访问接口
        public NewsManage()
        {
            ndao = new NewsDAO();
        }
        //选择全部新闻逻辑方法定义
        public DataTable SelectAll()
        {
                  //调用 NewsDAO 类的 SelectAll 方法返回查询
//DataTable 类型的结果
          return ndao.SelectAll();
          }
        //根据新闻 ID 取出该条新闻的具体内容的逻辑方法定义
        public News SelectById(string id)
        {
            return ndao.SelectById(id);
        }
            //其他逻辑方法略
    }
}
```

2）CategoryManage.cs (新闻类别管理业务逻辑类)。

3）CommentManage.cs (评论管理业务逻辑类)。

4）UsersManage.cs (用户管理业务逻辑类)。

6 UI 层 (用户界面) 的设计及对中间层的调用

6.1 添加网站

在解决方案上单击右键，执行 "添加-->新建网站-->ASP.NET 网站"，命名为 "新闻发布系统"。

6.2 添加动态网页

在网站中添加所需的 Web 窗体，生成有代码隐藏页的.aspx 文件，首页等前台页面直接放在网站根目录下，而后台管理相关网页，则放置于根目录下的 admin 文件夹中，以便于管理，项目结构如图 3 所示。以下列出部分代码页 (.cs) 文件中对中间层的调用代码：

(1) 根据选择的新闻 ID 取出该条新闻的具体代码：

```
//定义了 News 对象,因此在代码文件头部必须引用 Model
  using Model;
//在用户界面(UI)的隐藏代码中,都是靠创建 BLL 层对象来与
```

```
//数据库进行交互的,必须引用 BLL
  using BLL;
  //对象 n 必须定义为 public,否则.aspx 页面无法访问
    public News n = new News();
    //页面加载事件
  protected void Page_Load(object sender, EventArgs e)
  {
     string newsId;
 newsId = Request.QueryString["newsId"].ToString().Trim();
    // 构造 BLL 逻辑层的 NewsManage 对象 nm,实现对数据
//库的操作
      NewsManage nm = new NewsManage();
      //调用 nm 对象的 SelectById 逻辑方法,返回值为 News
//类的查找记录,赋给
//News 对象 n
     n=nm.SelectById(newsId);
      }
```

(2) 后台登录代码

```
  //登录按钮单击事件
protected void LoginButton_Click(object sender, EventArgs e)
  {
     //创建注册用户对象
  Users rs = new Users();
  //通过对象的属性(set)给对象的字段赋值
     rs.Uname = Login1.UserName;
     rs.Pwd = Login1.Password;
      //创建注册用户管理对象
     UsersManage um = new UsersManage();
      //如果用户名和对于密码存在于数据库中
     if (um.IsUserPwd(rs))
     {
     //    登录成功,创建 Session 变量
        Session["username"] = Login1.UserName;
        Response.Redirect("manage.aspx");
     }
     else
       //验证失败,弹出提示对话框
     {
            Page.ClientScript.RegisterStartupScript    (Page.
GetType (), "message", "<script language='javascript' defer>
alert(' 用户名或密码错误,请重试! ');</script>");
     }
  }
```

(3) 用户注册代码

```
    //注册按钮单击事件
   protected void RegisterButton_Click (object sender,
EventArgs e)
   {
      bool flag = false;
      Users rs1 = new Users();
      rs1.Uname = uName.Text;
      UsersManage um1 = new UsersManage();
      if (pwd.Text ! = pwd1.Text)
      {
            Page.ClientScript.RegisterStartupScript    (Page.
GetType (), "message", "<script language='javascript' defer>
alert(' 两次密码不一致! ');</script>");
      }
//IsUser 方法用来判断要注册的用户名在数据库中是否已存在
      else if (um1.IsUser(rs1))
      {
            Page.ClientScript.RegisterStartupScript    (Page.
GetType (), "message", "<script language='javascript' defer>
alert(' 用户名已存在! ');</script>");
      }
      else
      {
         Users rs = new Users();
         rs.Uname = uName.Text;
         rs.Pwd = pwd.Text;
         rs.Email = email.Text;
          rs.Uauthority  =  Convert.ToInt32 (DropDownList1.
SelectedValue);
         UsersManage um = new UsersManage();
         flag=um.AddUser(rs);
         if (flag)
         {
            Page.ClientScript.RegisterStartupScript   (Page.
GetType (), "message", "<script language='javascript' defer>
alert(' 用户注册成功! ');</script>");
           //注册成功后延迟 5 秒自动返回登录界面
     Response.Write("5 秒后自动返回登录界面,请稍后...");
           string strRedirectPage = "login.aspx";
           string strRedirectTime = "5";
           string strRedirect = string.Format ("{0};url={1}",
strRedirectTime, strRedirectPage);
           Response.AddHeader("refresh", strRedirect);
         }
         else
         {
            Page.ClientScript.RegisterStartupScript   (Page.
GetType (), "message", "<script language='javascript' defer>
alert(' 注册失败! ');</script>");
         }
      }
  }
```

(4) 更新新闻代码

```
 //页面加载事件,显示当前新闻各字段取值
protected void Page_Load(object sender, EventArgs e)
{
     News n = new News();
     NewsManage nm = new NewsManage();
     n = nm.SelectById(Request["newsId"]);
```

```
        if (! IsPostBack) //首次载入页面时绑定所选新闻内容和
//页面控件
            {
            DataTable dt = new DataTable();
            CategoryManage cm = new CategoryManage();
            dt = cm.SelectAll();
        //DropDownList1 为显示新闻分类的下拉列表框控件
            DropDownList1.DataSource = dt;
            DropDownList1.DataTextField = "name";
            DropDownList1.DataValueField = "id";
            DropDownList1.DataBind();
            DropDownList1.SelectedValue = n.CaId;
            txtTitle.Text = n.Title;
          // content1 为 textarea 控件的 id 值
            content1.Value = n.Content;
        }
    }
//更新按钮单击事件
    protected void NewsUpdateButton_Click (object sender,
EventArgs e)
    {
        News n = new News();
        n.Id = Request["newsId"];
        n.Title = txtTitle.Text;
        n.CaId = DropDownList1.SelectedValue;
        n.Content =content1.Value;
        NewsManage nm = new NewsManage();
        if (nm.Update(n))
        {
            Response.Write("修改成功！");
        }
        else
        {
            Response.Write("修改失败！");
        }
    }
```

(5) 删除新闻代码

```
NewsManage nm = new NewsManage();
      //Delete 方法返回值为 bool 型的 true 或 false
if (nm.Delete(Request["newsId"]))
{
    Response.Write("删除成功！");
}
else
{
    Response.Write("删除失败！");
}
```

7 结语

(1) 三层架构层间关系。

UI 层向上直接与用户进行交互，向下则通过访问 BLL 层(业务逻辑层) 实现具体业务，而 BLL 层只是逻辑上的功能定义，具体功能是靠 BLL 层向下访问 DAL (数据访问层) 来实现的，即只有 DAL 层是直接访问数据库的一层。而 Model 层(实体层) 则为各个层实现以对象的方式表现数据提供了支持。

(2) 数据库的删除触发器。

在系统中删除新闻和删除新闻类别时都会产生级联式删除，为此采用了 instead of delete 触发器 (trigger)，如：要删除某条新闻，先要删除对这条新闻的评论；同样，删除新闻类别，要先删除此类别新闻的所有评论，然后再删除所有此类新闻，最后再删除此类别。本系统开始分别添加了两个 instead of 触发器，来代替新闻和新闻类别的 delete 操作，后来发现删除类别时，对此类别的新闻删除不彻底，分析是由于两个 instead of delete 触发器产生了嵌套：删除类别触发器(trigCategoryDelete) 中嵌套了删除新闻触发器 (trigNewsDelete)，最后通过修改被嵌套的删除新闻触发器 (trigNewsDelete) 解决。关键代码如下：

1)修改前：delete from news where id=@newsId

2)修改后：delete from news where id in (select id from deleted)

(3) 数据库中字段取值是否为空的判断。

判断数据库 News 表的记录中 content 字段值是否为空，正确语句应为：if (dt.Rows [0] [" content"] == DBNull.Value)，初学者很容易错写成：if (dt.Rows [0] [" content"] == null)，错误原因：dt.Rows [0] [" content"] 得到的是记录中的 content 字段值，是值类型；而 null 属于引用类型，两者不会相等，比较结果永远为 false。

图 3 解决方案资源管理器中的项目

参考文献

[1] 王翔. ASP.NET 4.0 网站建设基础教程. 北京：北京邮电大学出版社，2012.

[2] 马骏. C# 程序设计及应用教程 2 版. 北京：人民邮电出版社，2009.

(收稿日期：2012-10-10)

人事档案管理系统的设计与实现

畅育超

摘　要：主要介绍利用 VB 实现人事档案管理系统的设计与实现。

关键词：ADO 记录集；树形控件；网格控件；事件代码

1　引言

随着社会的不断发展，各企事业单位对本单位人事信息的处理已呈现信息化，以便能及时动态地了解本单位人事信息。本系统主要用于解决企事业单位中员工的一些基本信息的存储、修改、统计、打印、查询和浏览等，目标是使人事档案管理真正实现实时化、准确化、无纸化。

2　功能模块

该系统主要由系统管理模块、报表打印模块、档案操作模块、人事变动模块、统计报表模块、数据维护模块 6 部分构成，在操作上能够实现诸如添加、修改、删除、按条件查询、新用户的设置及密码修改等方面的工作，基本满足人事档案管理的日常业务需要，如图 1 所示。

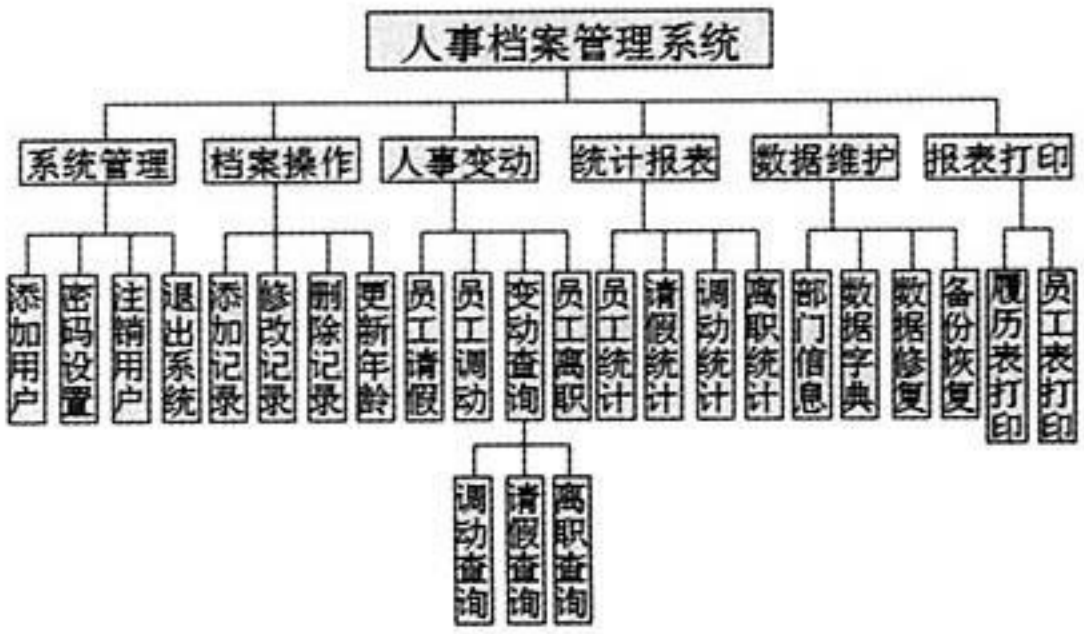


图 1　系统功能

3　具体实现

（1）公共变量及模块（Modbas）

```
Public s As Integer ′消息参数
Public c As String
Public b As String
Public Strs As String
Public Str As String
Public sq1 As Date ′定义字符型日期
Public sq2 As Date
Public CurrentUser As String ′当前用户参数
Public selrsid As String
Public selrsname As String
Public selrslid As String ′请假员工 id 参数
Public cxrs As New ADODB.Recordset ′员工状态查询记录
Public rslcx As New ADODB.Recordset ′员工请假查询记录
Public rslbookmark As Variant ′员工请假查询书签
Public rsbookmark As Variant ′员工状态查询书签
Public JbLen As Integer ′定义每级的长度
Public conn As New ADODB.Connection
Public a As String ′定义一个获取当前所选择的结点的 KEY 值
'Public Sub main()'数据库连接模块
conn.Open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=
" + App.Path + "\rsdata.mdb" + ";Persist Security Info=False"
 JbLen = 2 ′初始化为 2
 conn.CursorLocation = adUseClient'设置游标类型为客户端
End Sub
Public Sub cmddata_Click(rsgrid As MSFlexGrid) ′数据导出
′模块
Dim xlsRowCount As Integer, xlsColCount As Integer ′生成
′的表格的行数和列数
Dim xlsApp As Excel.Application
Dim xlsBook As Excel.Workbook
Dim xlsSheet As Excel.Worksheet
Dim i, j As Long
Set xlsApp = CreateObject("Excel.Application")
Set xlsBook = xlsApp.Workbooks.Add
Set xlsSheet = xlsBook.Worksheets(1)
On Error Resume Next
xlsRowCount = rsgrid.Rows
xlsColCount = rsgrid.Cols
With xlsSheet
For i = 1 To xlsRowCount - 1 ′设置行高
.Rows(i).RowHeight = 15
Next
′把 MSFlexGrid1 的内容写入到电子表格中
For i = 0 To xlsRowCount - 1
For j = 0 To xlsColCount - 1
.Cells(i + 1, j + 1).Value = rsgrid.TextMatrix(i, j + 1)
```

```
.Cells.EntireColumn.AutoFit ' 自动适应字段宽度
.Cells.HorizontalAlignment = Center ' 居中对齐
Next
Next
End With
xlsApp.Visible = True ' 显示电子表格
xlsBook.SaveAs App.Path & "\导出数据.xls"
Set xlsApp = Nothing ' 交还控制给 Excel
End Sub
```

(2) 用户登录模块

如图 2 所示。

图 2 用户登录界面

1) 用户身份验证过程:

① 用户登录窗体启动，输入用户名和密码。

② 系统检查操作员表是否有相应的用户名和密码。

③ 如果用户名或密码错误，系统有错误消息提示。

④ 如果用户名或密码同时正确，登录窗口本身卸载，主控平台窗口显示。

2) 主要代码:

```
Private Sub cmdok_Click()'确定按钮代码
Set rt = New ADODB.Recordset
rt.Open "select * from pass where username ='" & Trim
(txtusername.Text) & "' And userpassword ='" & Trim
(txtpassword.Text) & "'", conn, adOpenKeyset, adLockReadOnly
If rt.RecordCount > 0 Then
CurrentUser = Trim(txtusername.Text)
rt.Close
Set rt = Nothing
Unload Me
rsadmin.Show
Else
MsgBox " 用户名或密码错误，请重新输入！", vbOKOnly +
vbInformation, "警告"
txtusername.Text = ""
txtpassword.Text = ""
txtusername.SetFocus
End If
End Sub
```

(3) 系统主控平台模块

如图 3 所示。

图 3 系统主控界面

为了使设计直观、简单和方便用户操作，主控制平台窗口采用菜单栏和工具设计，树形控件 (Treeview) 便于显示部门信息，网格控件 (Msflexgrid) 用于浏览员工的基本信息和显示查询结果，在此界面中用户可进行数据浏览、查询、导出和打印操作，代码如下:

```
Private Sub Com3_Click() ' 按照员工状态(在职、退休等)查询
Dim sy As String
Dim i As Integer
sy = Trim(com3.Text)
If cxrs.State = 1 Then
cxrs.Close
End If
If sy = "所有员工状态" Then
cxrs.Open "select * from rsusertab order by rsid", conn,
adOpenKeyset, adLockOptimistic
Else
cxrs.Open "select * from rsusertab where rsstate='" & sy
& "' order by rsid", conn, adOpenKeyset, adLockOptimistic
End If
Private Sub Comfind_Click() ' 按姓名、年龄、职称等字段进行综
合查询
com3.ListIndex = 1
If txt1.Text = "" Then
MsgBox "请输入查询内容！", vbInformation, "查询提示"
Exit Sub
End If
Dim str1 As String
Dim str2 As String
Dim str3 As String
Dim querystr As String
Dim str4 As String
Dim i As Integer
str1 = Trim(com1.Text)
Select Case str1
Case "姓名"
    str2 = "rsname"
Case "性别"
    str2 = "rssex"
Case "年龄"
    str2 = "rsold"
```

```
Case "政治面貌"
    str2 = "rszzm"
Case "本单位工龄"
    str2 = "rsnewyear"
Case "文化程度"
    str2 = "rseducation"
Case "职称"
    str2 = "rsprofess"
End Select
str4 = ""
querystr = Trim(txt1.Text)
If str1 = "年龄" Or str1 = "本单位工龄" Then
querystr = CStr(Trim(txt1.Text))
str4 = ""
End If
If cxrs.State = 1 Then
cxrs.Close
End If
If Trim(com2.Text) = "Like" Or Trim(com2.Text) = "Not Like"
Then
cxrs.Open "select  *  from rsusertab  where  " & str2 & " " &
Trim  (com2.Text) & " '%" & querystr & "%' and rsstate='" &
Trim  (com3.Text) & "'  order by rsid", conn, adOpenKeyset,
adLockOptimistic
 Else
cxrs.Open "select  *  from rsusertab  where  " & str2 & " " &
Trim  (com2.Text) & " " & str4 & "" & querystr & "" & str4 & "
and  rsstate='" & Trim  (com3.Text) & "' order by rsid", conn,
adOpenKeyset, adLockOptimistic
End If
If cxrs.EOF Then
    MsgBox "没有找到符合条件的记录！", vbInformation, "查询
提示"
msfgrid.Enabled = False
txt1.Text = ""
Exit Sub
Else
    Label3.Caption = " 满足条件的记录有" & cxrs.RecordCount
& "条"
msfgrid.Enabled = True
With msfgrid
.Rows = 1
Do While Not cxrs.EOF
.Rows = .Rows + 1
.TextMatrix(.Rows - 1, 0) = .Rows - 1
For i = 0 To cxrs.Fields.Count - 1
.Col = i + 1
If IsNull(cxrs.Fields(i)) Then
.TextMatrix(.Rows - 1, i + 1) = Empty
Else
.TextMatrix(.Rows - 1, i + 1) = cxrs.Fields(i)
End If
Next i
cxrs.MoveNext
Loop
End With
End If
txt1.Text = ""
End Sub
Private Sub msfgrid_Click()' 网格控件(Msflexgrid)的单击模块
selrsid = msfgrid.TextMatrix(msfgrid.Row, 1)
selrsname = msfgrid.TextMatrix(msfgrid.Row, 2)
cxrs.MoveFirst ' 重新定位记录指针位置
cxrs.Find "rsid=" & selrsid & "" ' 查找当前选定的记录
rsbookmark = cxrs.Bookmark ' 记忆书签
End Sub
Sub delrsado() ' 删除记录模块
If MsgBox (" 确认要删除工号为" & selrsid & " 的记录吗？",
vbInformation + vbYesNo, "温馨提示") = vbYes Then
   conn.Execute "delete from rsusertab where rsid ='" &
selrsid & "'"
   Else
    Exit Sub
End If
End Sub
Private Sub trview_Click()'树形控件(treeview)单击模块
If Not trview.SelectedItem Is Nothing Then
   selitem = trview.SelectedItem.Text
   If cxrs.State = 1 Then
   cxrs.Close
   End If
   cxrs.Open "select * from rsusertab where rsdepartment='"
& selitem & "' and  rsstate='" & Trim  (com3.Text) & "' order
by rsid", conn, adOpenKeyset, adLockOptimistic
   If cxrs.RecordCount > 0 Then
   Label3.Caption = " 满足条件的记录有" & cxrs.RecordCount
& "条"
   msfgrid.Enabled = True
   With msfgrid
.Rows = 1
Do While Not cxrs.EOF
.Rows = .Rows + 1
.TextMatrix(.Rows - 1, 0) = .Rows - 1
For i = 0 To cxrs.Fields.Count - 1
.Col = i + 1
If IsNull(cxrs.Fields(i)) Then
.TextMatrix(.Rows - 1, i + 1) = Empty
Else
.TextMatrix(.Rows - 1, i + 1) = cxrs.Fields(i)
End If
Next i
cxrs.MoveNext
Loop
End With
```

```
Else
msfgrid.Enabled = False
End If
  End If
End Sub
```

(4) 系统管理模块

系统管理模块主要由图 4 的两个子模块构成，主要功能是完成管理员基本信息的建立、编辑和删除。

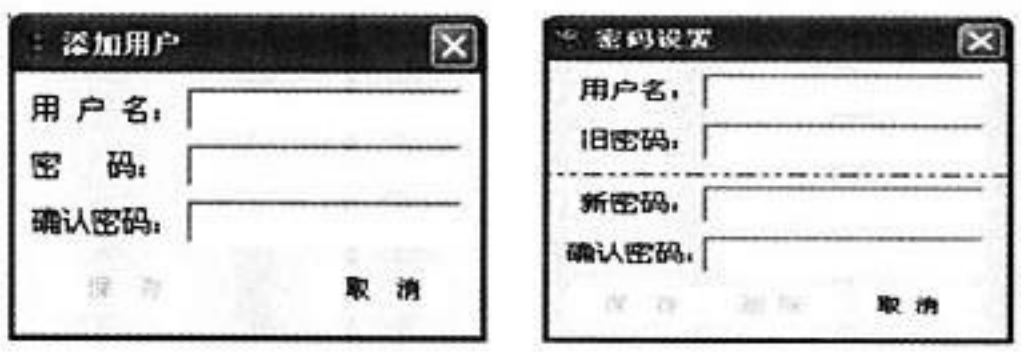


图 4

1) 增加新管理员过程

① 启动添加管理员窗口，输入用户名、密码和确认密码。

② 如果两次输入密码不一致，系统有错误信息提示。

③ 确认操作。

④ 如果用户名、密码和确认密码正确，则完成对数据库操作员表的操作。

2) 修改操作员密码过程

① 启动操作员密码设置窗口，输入用户名和旧密码。

② 如果用户名和旧密码错误，系统有错误信息提示，则不能进行修改密码和删除管理员操作。

③ 确认操作。

④ 如果用户名和旧密码正确，修改密码和删除管理员都可进行，则完成对数据库操作员表的相应操作。

3) 主要代码

```
  Private Sub shanchu_Click()'删除用户模块
Dim pls As String
Set ps = New ADODB.Recordset
pls = "select * from pass where username ='" & Trim
(txtusername.Text) & "' and userpassword ='" & Trim
(txtoldpwd.Text) & "' order by username"
ps.Open pls, conn, adOpenKeyset, adLockOptimistic
If ps.RecordCount > 0 Then
 If MsgBox("您真的要删除该用户吗？", 32 + 4, "信息确认") =
vbYes Then
 ps.Delete
 txtusername.Text = ""
 txtoldpwd.Text = ""
 txtusername.SetFocus
 End If
 Exit Sub
 Else
 MsgBox "该用户不存在或密码错误，请重新输入！",
vbInformation, "温馨提示"
 txtusername.Text = ""
 txtoldpwd.Text = ""
 txtusername.SetFocus
 End If
 ps.Close
End Sub
Private Sub txtnewpwd1_LostFocus()'密码验证
If txtnewpwd.Text <> txtnewpwd1.Text Then
MsgBox "两次输入密码不一致，请重新输入！", vbInformation,
"温馨提示"
txtnewpwd.Text = ""
txtnewpwd1.Text = ""
txtnewpwd.SetFocus
End If
End Sub
```

(5) 档案操作模块

如图 5 至图 7 所示。

该模块的主要功能是完成员工基本信息的添加、修改、浏览和打印，并且可以根据员工的身份证号进行批量更新员工年龄、性别和出生日期。

1) 增加员工基本信息过程

① 启动添加员工窗口，输入或修改员工的基本信息，其中工号、年龄、工龄、本单位工龄和登记人员信息由系统自动生成，员工信息可以连续修改。

② 如果输入或修改的信息不合法或为空值，系统有错误信息提示。

③ 确认操作。

④ 如果员工的基本信息正确，则完成对数据库员工表的操作。

2) 浏览员工基本信息过程

启动浏览员工信息窗口，使用“最前”、“上一条”、“下一条”、“最后”4 个命令按钮，即可逐条浏览员工基本信息。

3) 打印员工履历表

启动浏览员工信息窗口，选择要打印的员工履历表，使用“打印”按钮，即可打印或预览当前选定的员工履历表。

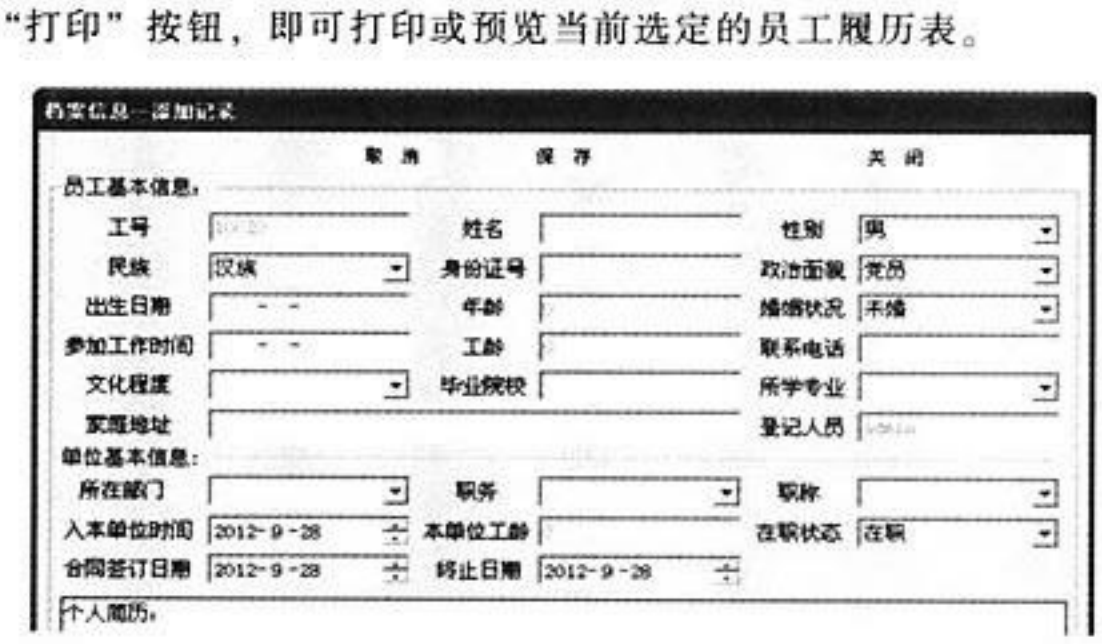


图 5 添加记录

图 6　浏览、修改记录

4) 按“员工身份证号码”对员工性别、年龄等信息进行更新:

```
Private Sub uodateold_Click(Index As Integer)'员工信息更新模块
Dim upado As New ADODB.Recordset
upado.Open " select  * from rsusertab order by rsid ", conn, adOpenDynamic, adLockOptimistic
  Do While Not upado.EOF
 upado.Fields("rssex") = XB(upado.Fields("rsdisid")) ' 更新性别
    upado.Fields("rsbrith") = Csnydata(upado.Fields("rsdisid")) ' 更新出生年月
    upado.Fields ("rsold") = Int (DateDiff ("y", upado.Fields("rsbrith"), Date) / 365.25) ' 更新年龄
    upado.Update
    upado.MoveNext
  Loop
upado.Close
Set upado = Nothing
Call Com3_Click ' 刷新网格
MsgBox " 按身份证号对性别、出生日期、年龄信息已成功更新！ ", vbInformation, "温馨提示"
End Sub
```

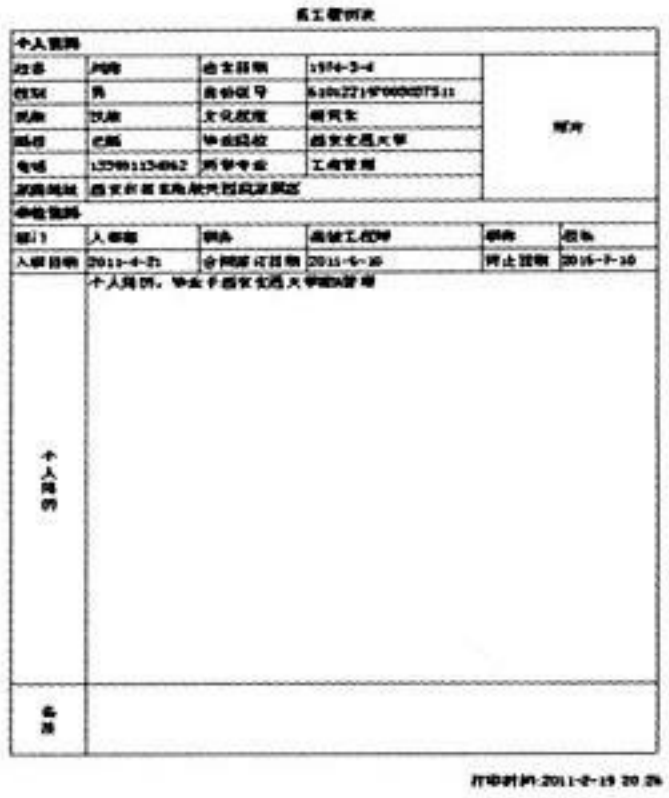


图 7　报表打印

(6) 人事变动模块

如图 8 至图 10 所示。

图 8　请假、调动信息添加

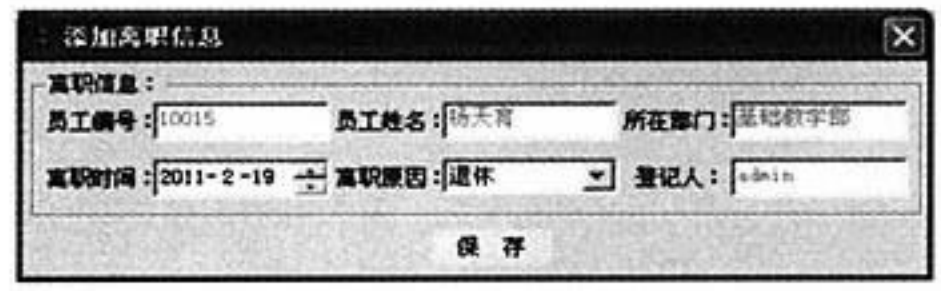


图 9　添加离职信息

1) 添加员工请假信息

① 首先在主控平台选定要请假的员工。

② 如果员工在休假，必须先销假，否则不能再进行请假。

③ 启动添加员工请假窗口，输入相关信息。

④ 如果信息正确，则完成对数据库员工请假表的操作。

2) 添加员工调动信息

① 首先在主控平台选定要调动的员工,。

② 启动添加员工调动信息窗口，输入相关信息。

③ 如果信息正确，则完成对数据库员工调动表的操作。

3) 添加员工离职信息

① 首先在主控平台选定要调动的员工,。

② 启动添加员工离职信息窗口，输入相关信息。

③ 如果信息正确，则完成对数据库员工离职表的操作。

图 10　信息编辑

4) 员工请假信息编辑

① 首先在员工请假查询窗口输入员工姓名进行查询。

② 如果有记录，则可进行修改、删除、销假等操作，否则系统有错误提示。

③ 确认操作，如果信息正确，则完成对数据库员工请假表的操作。

5) 员工调动信息编辑

① 首先在员工调动查询窗口输入员工姓名进行查询。

② 如果有记录，则可进行删除、浏览和导出操作，否则系统有错误提示。

③ 确认操作，则完成对数据库员工调动表的操作。

6) 员工离职信息编辑

① 首先在员工离职查询窗口输入员工姓名进行查询。

② 如果有记录，则可进行删除、浏览和导出操作，否则系统有错误提示。

③ 确认操作，则完成对数据库员工离职表的操作。

7) 主要代码

```
Private Sub rlsfgrid_Click()'网格控件单击模块
selrslid = rlsfgrid.TextMatrix(rlsfgrid.Row, 1)
rslcx.MoveFirst ' 重新定位记录指针位置
rslcx.Find "rslid='" & selrslid & "'" ' 查找当前选定的记录
rslbookmark = rslcx.Bookmark ' 记忆书签
End Sub
Private Sub cancels_Click() ' 销假模块
If rslcx.EOF = True Then
rslcx.MoveLast
End If
rslcx.Fields("rslstate") = "已销假"
rslcx.Update
MsgBox "销假成功！", vbInformation, "温馨提示"
cancels.Enabled = False
rlsfgrid.Enabled = False ' 网格
End Sub
```

（7）统计报表模块

如图 11 所示。

图 11 统计报表

以下 4 个窗口主要用于显示新进员工、员工请假、员工离职、员工调动信息，同时可以将统计结果进行导出。

1) 新进员工统计

① 首先设定时间段。

② 确认操作，如果有记录，则显示记录，同时将统计结果导出，否则无显示。

2) 员工请假统计

① 首先设定时间段。

② 确认操作，如果有记录，则显示员工请假的次数、总天数等信息，同时可将统计结果导出，否则无显示。

3) 员工调动、离职统计

① 首先设定时间段。

② 确认操作，如果有记录，则显示记录，同时将统计结果导出，否则无显示。

4) 主要代码

```
Private Sub Form_Load()'员工请假情况统计模块
Call rsldg1ref ' 生成表头
totalout.Enabled = False
Dim rsltotal As New ADODB.Recordset
rsltotal.Open "select rslid, rslname,(select count (rslid) from rsleave where leavedate between " & Chr (35) & sq1 & Chr (35) & " and " & Chr (35) & sq2 & Chr (35) & " and rslid=a.rslid),sum (leaveday) from rsleave a where leavedate between " & Chr(35) & sq1 & Chr(35) & " and " & Chr(35) & sq2 & Chr (35) & " group by rslname,rslid ", conn, adOpenKeyset, adLockReadOnly
If rsltotal.RecordCount > 0 Then
labtxt1.Caption = Format(sq1, "yyyy 年 mm 月 dd 日") & "至" & Format(sq1, "yyyy 年 mm 月 dd 日") & "请假统计"
totalout.Enabled = True
With rlsfgrid
.Rows = 1
Do While Not rsltotal.EOF
.Rows = .Rows + 1
.TextMatrix(.Rows - 1, 0) = .Rows - 1
For i = 0 To rsltotal.Fields.Count - 1
.Col = i + 1
If IsNull(rsltotal.Fields(i)) Then
.TextMatrix(.Rows - 1, i + 1) = Empty
Else
.TextMatrix(.Rows - 1, i + 1) = rsltotal.Fields(i)
End If
Next i
rsltotal.MoveNext
Loop
End With
Else
MsgBox "没有符合条件的统计结果！", vbInformation, "温馨提示"
End If
rsltotal.Close
Set rsltotal = Nothing
End Sub
Private Sub new_1_Click(Index As Integer) ' 新进员工统计模块
seldate.Caption = "新进员工报表"
seldate.Frame1.Caption = "进入本单位时间:"
seldate.Show 1
If cxrs.State = 1 Then
cxrs.Close
End If
cxrs.Open "select * from rsusertab where rsnewdate between " & Chr(35) & sq1 & Chr(35) & " and " & Chr(35) & sq2 & Chr (35) & " and rsstate =' 在职 ' ", conn, adOpenKeyset, adLockOptimistic
```

```
If cxrs.RecordCount > 0 Then
Label3.Caption = "满足条件的记录有" & cxrs.RecordCount & "条"
msfgrid.Enabled = True
With msfgrid
.Rows = 1
Do While Not cxrs.EOF
.Rows = .Rows + 1
.TextMatrix(.Rows – 1, 0) = .Rows – 1
For i = 0 To cxrs.Fields.Count – 1
.Col = i + 1
If IsNull(cxrs.Fields(i)) Then
.TextMatrix(.Rows – 1, i + 1) = Empty
Else
.TextMatrix(.Rows – 1, i + 1) = cxrs.Fields(i)
End If
Next
cxrs.MoveNext
Loop
End With
Else
msfgrid.Clear
Call dg1ref ' 调用表头
msfgrid.Rows = 1
msfgrid.Enabled = False
Label3.Caption = "满足条件的记录有 0 条"
End If
End Sub
```

(8) 数据维护模块

如图 12 所示。

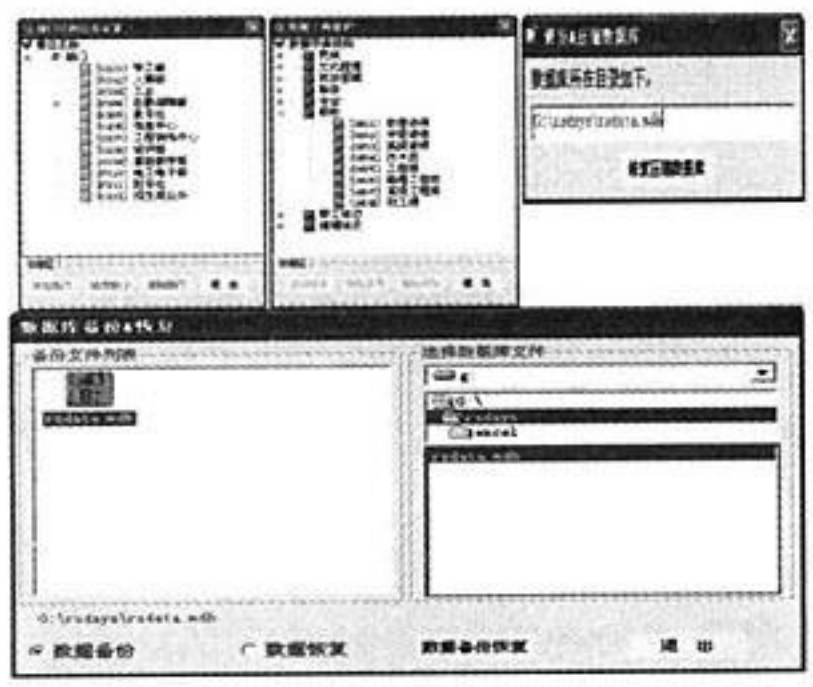


图 12 数据维护

该模块主要完成部门信息、数据字典的数据添加、删除、修改和浏览、数据库修复、压缩、备份和恢复功能。

1) 部门信息或数据字典信息编辑

① 首先确认数据库的名称和所在目录是否正确。

② 进行添加、修改、删除操作，（不能直接修改或删除有子项的父项）。

③ 确认操作，如果信息正确，则完成对数据库部门表或数据维护表的操作。

2) 数据库修复压缩

① 首先确认数据库名称和所在目录是否正确。

② 如果正确，确认操作。

③ 完成对数据库的修复与压缩操作。

3) 数据库备份

① 启动数据库备份与恢复窗口，选定要备份的数据库。

② 确认操作，完成对数据库的备份，备份文件将显示在“备份文件列表”中。

4) 数据库恢复

① 启动数据库备份与恢复窗口，在“备份文件列表”中选定要恢复的数据库备份文件。

② 确认操作完成，对数据库的恢复。

5) 数据备份恢复主要代码

```
Private Sub Command1_Click()
If conn.State = adStateOpen Then'判断数据库是否已经打开
  conn.Close
End If
rsadmin.msfgrid.Enabled = False
If Option1.Value = True Then
  If File1.ListCount <> 0 Then
FileCopy Trim(Label2.Caption), Date & File1.FileName & "bak"
   Me.MousePointer = 0
   MsgBox "数据已经备份完成", vbInformation, "温馨提示"
   key = Date & File1.FileName & "bak"
   Set itmX = ListView1.ListItems.Add(, , key, 1)
  End If
End If
If Option2.Value = True Then
 If File1.ListCount <> 0 Then
   FileCopy ListView1.SelectedItem, File1.FileName
   Me.MousePointer = 0
   MsgBox "数据已恢复完毕！"
  Else
   MsgBox "请选择要恢复的数据", vbInformation, "温馨提示"
  End If
 End If
End Sub
```

4 结语

本系统具有功能相对完备、操作简单方便等特点，可以实现数据输入、修改、查询、浏览、统计和打印功能，极大地提高企事业单位人事档案的管理效率和水平；本系统尚待改进和增添的部分功能，如界面的美观、布局，导出数据需要改进，也需要增加员工的培训、奖惩、家庭社会关系、职称晋级、请假条打印等模块，使系统功能更完善、更合理。

（收稿日期：2012-09-28）

在线生成 Word 文档的实现与应用

汪永松

摘　要：从开发者的角度介绍了在 B/S 架构下实现在线 Word 文档的生成，并通过开发实例对此应用进行详细说明。读者不仅可以了解在 B/S 平台下实现在线文档生成的开发过程，而且还能体会到办公组件开发的技巧和乐趣。

关键词：在线文档生成；B/S 架构；COM 组件；JACOB 开源项目

1　概述

1.1　在线生成 Word 文档应用实例

所谓在线生成 Word 文档就是在网页上控制 Word 文档的生成，而生成后的 Word 文档经过下载后即可作为本地 Word 文档，访问者可以直接编辑和打印文档；而相比之下，大多数情况下，由于网页结构的复杂性（例如 DIV、IFRAME 等组件的嵌套），访问者无法实现直接对网页内容进行打印，或者将网页内容转换到本地 Word 文档。

在线生成 Word 文档实际上是办公需求（办公文档处理）与 B/S 应用（门户网站等）的结合，其既可实现 B/S 应用下的共享方式，又延续了处理办公文档的习惯。

图 1 和图 2 分别是在线生成 Word 的网页内容和生成 Word 后的网页效果。

图 1　在线生成 Word 文档网页

图 2　在线 Word 文档生成页面

通过将图 2 与图 1 对比可知，在线生成的 Word 文档内容与网页同步，且格式和排版已经制定，用户无需再花费过多精力进行文档的处理，因此提高了文档处理的效率。

1.2　实现要点

实际上，在线生成 Word 文档在内容展现形式上结合了 B/S 架构特性（例如：基于 HTTP 的内容推送、OLE 对象的嵌入），而其后台处理类似开发者熟知的单机应用，其实质上还是对 Office COM 组件的应用。

通过 Office COM 组件，开发者可以方便地对 Office 文档进行操作，例如：新建、保存、另存为、选取文本块、插入文本等。使用 Office COM 组件即可实现 Word 文档的生成。而要实现文档的在线生成，一则需要 B/S 服务端后台使用 Office COM 组件，另则需要有数据源支撑所生成文档的内容的动态性。在线 Word 文档生成示意如图 3 所示。

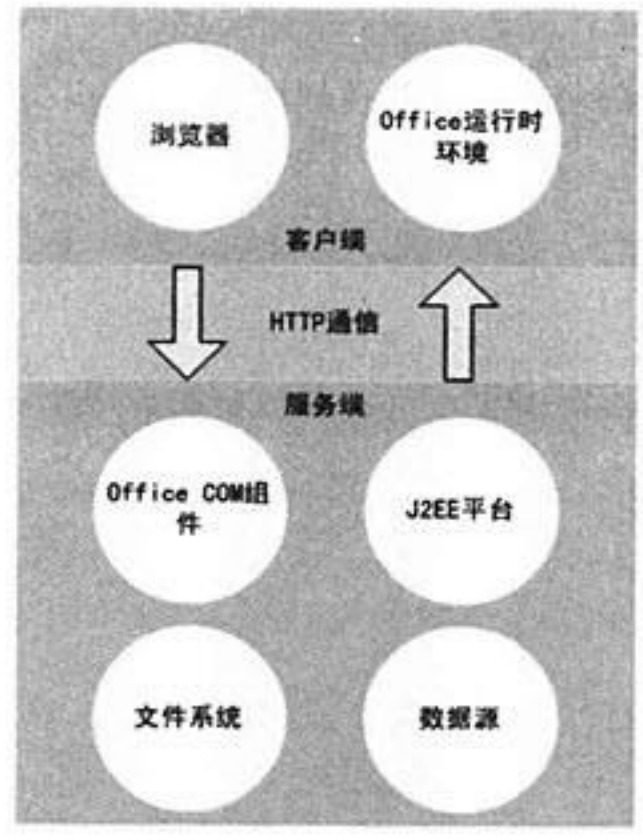


图 3　在线 Word 文档生成示意

图 3 中，Web 服务端采用的是 J2EE 平台，则其数据源提取将会采用 JDBC 方式。由此可见，在线生成 Word 文档的关键点：Word 程序 COM 组件应用、文档内容提供和页面交互。

1.2.1 Word 程序 COM 组件

Word 程序 COM 组件是生成 Word 文档的核心，而对于 COM 组件的应用，大多数开发者的开发环境恐怕还是以微软

公司的 VS 或宝兰公司的 C++ Builder 或 Delphi 开发环境为主。对于 Java 开发者，接触 COM 组件的机会显然要比 C++程序员要少得多。那么如何让 Java 程序员用上 Word 程序的 COM 组件呢？

鉴于数量庞大的 Java 程序员和 Office 用户群体，该问题很快得到了较好地解决，一个叫 JACOB（JAVA-COM Bridge）的开源项目提供了桥接 Java 到 COM 组件的库，Java 程序员可以借助该库来调用 COM 组件。表 1 中是该库中重要类的及说明。

表 1　JACOB 库中重要类及说明

类/接口	说明
com.jacob.activeX.ActiveXComponent	ActiveX 组件，表示 COM 组件对象
com.jacob.com.Dispatch	调度接口，用于调用接口方法
com.jacob.com.Variant	多格式数据类型，作为接口方法的参数类型

需要注意的是，JACOB 通过 Java 本地化接口（JNI）的方式来实现 Java 到 COM 的桥接，所以其部署时还需要同时分发本地库（动态链接库，即 DLL 文件）。

1.2.2 文档内容

结合笔记的开发经验，文档内容的提供可分为两个部分：第一是文档模板；第二是后台数据。

（1）文档模板

文档模板可以理解为文档中模式相对固定的内容，例如：页眉、标题以及内容标签，如图 4 所示的就是本案例中的文档模板。

在图 4 中的模板中，文档的标题文字和表格的头部都是固定内容，而表格的单元格采用的是描述内容的标签，其格式为：[%=RS (数据集 ID,行号,列号) %]，这些标签最终将要被实际数据所替换。

图 4　WORD 模板文件

（2）后台数据

后台数据即从数据库中所获取的内容，JSP 页面从数据库获取数据后，利用 Word 程序 COM 组件的替换方法将后台数据替换文档模板中的标签。

1.2.3 页面交互

JSP 页面中，前端传给后端的内容主要是文档模板编号，即替换哪个文档模板；后端推送给前端的内容主要是生成后的文档。

2　设计过程

2.1　实现思路

在线生成 Word 文档应用的大致思路如下：

（1）编制文档模板，定义模板中标签内容与后台数据项的对应关系（示例中采用的是数据集的行列对应）。

（2）当用户点击在线生成 Word 时，JSP 页面前端将文档模板编号传入后端。

（3）服务端后端程序按照文档模板编号获取对应的数据源 ID 列表，再按照各个数据源通过 JDBC 方式获取数据集。

（4）服务端后端程序按照文档模板编号通过 Word 程序 COM 组件的读取文档模板并遍历其中需要替换的内容标签，并使用标签对应的后台数据替换标签文本。

（5）服务端后台程序将处理完毕的文档模板另存为 Word 文档。

（6）JSP 页面将保存的 Word 文档路径通过 HTML 链接推送给前端页面。

（7）用户通过点击 HTML 链接来下载或使用浏览器打开 Word 文档。

2.2　主要类设计

2.2.1 文档生成器—DocBuilder

顾名思义，文档生成器主要用于在后台生成 Word 文档。其实现 2.1 节中第（3）到（5）的功能。其关键成员有 3 个：

（1）文档模板 ID，用于接收 JSP 页面传过来的文档模板标识。

（2）Word 文档工具，用于获取文档模板中的所有需要替换的文本标签、使用后台数据替换文本标签和将替换后的文档另存为新的 Word 文档。

（3）数据集容器，其是一个以数据集名为键的散列表容器，其元素值是一个二维数组，用于存放结果集的单元值。

2.2.2 Word 工具类—MsWordUtil

Word 工具类是在 JACOB 的基础上对 Word COM 对象的操作所进行的再次封装，其以单例的形式提供了对 Word 文档的操控，其主要包括：

（1）建立/断开与 Word 程序 COM 组件的连接。

（2）文档操作，包括：新建、保存、另存和关闭。

（3）获取文档中所有指定左右标签的文本块。

（4）使用新文本替换文档中所有旧文本。

2.2.3 标签解析器—TagParser

标签解析器用于解析文档模板中文本标签内的文本，如图 4 所示的文档模板中，表格单元格中处于左右标签“[%”和

“%]”内的文本：“=RS (数据集 ID,行号,列号)”。

每解析一个文本标签即可检索其对应的后台数据，并进行文本替换，例如将标签“[%=RS (1,0,1) %]”替换成结果集 1 中第 1 行第 2 列的列值“张三”。

2.2.4 Web 端生成在线文档的桩模块—stub.jsp

该 JSP 页面用于接收前端页面传递的文档模板参数，并调用后台的文档生成器来生成目标 Word 文档，并以 HTML 链接的方式将目标文档的 URL 推送到前端页面。

2.3 数据库设计

案例中使用的是 Oracle 数据库，其中示例数据表是员工表(TAB_STAFF)，其结构定义如表 2 所示。

表 2 员工表规格定义

字段名	类型	约束	默认值	备注
STA_ID	VARCHAR2 (32)	PK		工号
STA_XM	VARCHAR2 (20)			姓名
STA_XB	VARCHAR2 (16)		FK	性别
STA_SR	DATE			出生日期
STA_RZRQ	DATE			入职日期
STA_ZGXL	VARCHAR2 (16)		FK	最高学历
STA_BYXX	VARCHAR2 (50)			毕业学校
STA_ZP	BLOB			照片
STA_BM	VARCHAR2 (16)		FK	部门
STA_GW	VARCHAR2 (16)		FK	岗位

3 开发过程

3.1 Web 端在线文档生成桩模块

3.1.1 在线文档生成桩模块头部定义

在线文档生成的桩模块是 JSP 页面，其需要调用后台的文档生成器对象，所以需要在 JSP 头部对所包含的库进行定义，JSP 页面的头部定义如代码 1 所示。

代码 1 在线文档生成桩模块的头部定义

文件名:stub.jsp

```
<%@ page language="java" import="java.io.*,foolstudio.bi.*"
pageEncoding="gb2312"%>
<%
    final String path = request.getContextPath();
    final String basePath = request.getScheme() + "://" +
request.getServerName()+
":"+request.getServerPort()+path+"/";
    final String baseRealPath = this.getServletContext().
getRealPath("/");
%>
```

需要注意的是，包含代码 1 中“foolstudio.bi”包的 Jar 文档需要分发在当前域的“WEB-INF\lib”文件夹中，另外 JACOB 相关的库文件也必须分发在该文件夹中。

3.1.2 在线文档生成桩模块的主体

代码 2 中是在线文档生成桩模块的主体代码，其中主要是调用后台的文档生成器对象，生成目标文档，并将其 URL 推送到前端页面。

代码 2 在线文档生成桩模块的主体代码

文件名:stub.jsp

```
<%
    String template = request.getParameter("template");

    if(template == null) {
        out.println("<p>无效的文档模板！</p>");
        return;
    }
    response.setCharacterEncoding("gb2312");
    response.setContentType("text/html");
    String templatePath = baseRealPath + "templates/";
    String outputPath = baseRealPath + "temp/";
    DocBuilder builder = new DocBuilder (template,
baseRealPath);
    builder.saveAs(templatePath, outputPath);
    String outputRealFile = outputPath+builder.getOutput();
String outputFile = basePath+"temp/"+builder.getOutput();
    File aFile = new File(outputRealFile);
    if(aFile.exists() ) {
        out.println("<p>Word 报告生成成功！</p>");
        out.println ("<p><a href=\"" + outputFile + "\">查看
或下载</a></p>");
    out.println ("<p><a href=\"#\" onclick=\"window.close();\"
>关闭本窗体</a></p>");
    }
    else {
        out.println("<p>抱歉！报告生成失败,请与管理员联
系！</p>");
     out.println ("<p><a href=\"#\" onclick=\"window.close();\"
>关闭本窗体</a></p>");
    }
%>
```

3.2 文档生成器

3.2.1 配置文件

代码 3 是文档生成器所依赖的配置文件，其中定义了数据库的连接信息、文档模板与数据源的对应关系和数据源的定义。

代码 3　配置文件
文件名：db.properties

```
URL=jdbc:oracle:thin:@localhost:1521/FOO
USER=foo
PASSWD=*********
staff_ds=ds1
ds1=select STA_ID,STA_XM,STA_XB,to_char(STA_SR,'yyyy-mm-dd') STA_SR,
to_char(STA_RZRQ,'yyyy-mm-dd') STA_RZRQ,
STA_ZGXL,STA_BYXX,STA_BM,STA_GW from tab_staff
```

3.2.2 文档生成器初始化

代码 4 是文档生成器初始化代码，其主要内容是根据文档模板获取对应的数据源列表，并将所有数据源中的数据读出并存放到数据集容器中。

代码 4　初始化文档生成器
文件名：DocBuilder.java

```
public DocBuilder(String templateId, String ctxPath) {
    this.mTemplateId = templateId;
    Properties props = new Properties();
    String dsList = null;
    try {
        props.load (new FileInputStream (ctxPath + DB_SETTING));
        dsList = props.getProperty (templateId + "_" + DS_PREFIX);
    } catch (FileNotFoundException e) {
        FooDebug.getInstance().println(e.getMessage());
        return;
    } catch (IOException e) {
        FooDebug.getInstance().println(e.getMessage());
        return;
    }

    if(dsList == null) {
        FooDebug.getInstance().println("无效的文档模板！");
        return;
    }
    //构造结果集容器
    this.mResuleSet = new HashMap <String,ArrayList < ArrayList<String>>>();

    try {
        //初始化结果集容器
        initResultSet(dsList, props);
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

代码 4 中，存放结果集的容器是一个以结果集名为键，以二维数组为值的散列容器，主要是考虑到一个文档模板中可能包含多个结果集。

3.2.3 初始化结果集容器

初始化结果集容器主要内容是根据数据源的定义从后台数据库中提取数据并填充到结果集容器中，代码 5 是初始化结果集容器的主要代码。

代码 5　初始化结果集容器
文件名：DocBuilder.java

```
//初始化结果集
private void initResultSet (String dsList, Properties props) throws SQLException {
    //获取数据库配置
    final String url = props.getProperty("URL");
    final String user = props.getProperty("USER");
    final String passwd = props.getProperty("PASSWD");
    if(url==null || user==null || passwd==null) {
        return;
    }
    //判断 Oracle 驱动是否存在
    if(FooDb.getInstance().isFoundOracleDriver()==false) {
        return;
    }
    Connection conn = FooDb.getInstance ().getConnector (url, user, passwd);
    Statement stat = FooDb.getInstance ().getQueryStat (conn);
    //分割所有数据源
    final String[] dss = dsList.split(",");
    //遍历所有数据源
    for(int i = 0; i < dss.length; ++i) {
        String sql = props.getProperty(dss[i]);
        ResultSet rs = FooDb.getInstance ().execQuery (stat, sql);
    int colsCount = FooDb.getInstance().getColsCount(rs);
        //定义单数据源的记录行
        ArrayList <ArrayList <String >> rows = new ArrayList<ArrayList<String>>();
        //表头各列
        /*
    ArrayList<String> header = new ArrayList<String>();
        for(int i = 1; i <= colsCount; ++i) {
    header.add(FooDb.getInstance().getColName(rs, i));
        }
        rows.add(header);
        */
        //遍历各行
        while(rs.next()) {
            ArrayList <String > row = new ArrayList < String>();
            for(int j = 1; j <= colsCount; ++j) {
                row.add(rs.getString(j));
            }
```

```
            rows.add(row);
        }
        FooDb.getInstance().closeRs(rs);
        //将单个结果集存入到结果集容器
        this.mResuleSet.put(dss[i], rows);
    }
    //关闭语句对象和数据库连接
    FooDb.getInstance().closeStat(stat);
    FooDb.getInstance().closeConn(conn);
}
```

代码 5 中，文档解析器分割指定文档模板所对应的数据源列表，然后逐一读取数据源中的记录行，最后将单个结果集存入到结果集容器中。

这种首先将所有数据源的数据加载到内存的方式可能会对系统内存的占用造成一定的压力，但是对于后面的数据检索而言，会显著提高检索效率。对于此处需要开发者按照实际的情况进行权衡。

3.2.4 另存文档

结果集容器初始化之后，要做的就是从文档模板中读取所有需要替换的标签文本，并一一与容器中的数据项进行对应，继而进行替换，追回将替换完毕的文档另存为新的 Word 文档。代码 6 是另存 Word 文档的主要代码。

代码 6　另存 Word 文档

文件名：DocBuilder.java

```
//读取模板和输出信息,调用保存模块
public void saveAs(String tempPath, String outPath) {
    __saveAs(tempPath+getTemplate(), outPath+getOutput
() );
}
//替换模板并保存输出
private void __saveAs (String tempFilePath, String
outFilePath) {
    if(mWordUtil.connect(false) == false) {
        return;
    }
    //打开文档模板
    if(mWordUtil.open(tempFilePath) == false) {
        mWordUtil.disconnect();
        return;
    }
    //获取文档模板中所有标签文本
    HashSet <String > texts = mWordUtil.getTagText
(TAG_LEFT, TAG_RIGHT);
    if(texts == null) {
        mWordUtil.close();
        mWordUtil.disconnect();
        return;
    }
    //解析标签文本并进行替换
    Iterator<String> it = texts.iterator();
    while(it.hasNext() ) {
        TagParser parser = new TagParser(it.next() );
        switch(parser.getType() ) {
            case TagParser.TAG_TYPE_RS: {
                handleRsTag(parser);
                break;
            }
            default: {
                break;
            }
        }
    }
    //另存文档
    try {
        mWordUtil.saveAs(outFilePath);
        mWordUtil.close();
        mWordUtil.disconnect();
    } catch(Exception e) {
        e.printStackTrace();
        return;
    }
    FooDebug.getInstance().println("文档处理完毕！ ");
}
```

3.2.5 替换标签文本

代码 7 是替换标签文本的关键代码。

代码 7　替换标签文本

文件名：DocBuilder.java

```
//处理数据集标签(本数据集外)
private void handleRsTag(TagParser parser) {
    //获取标签文本中的数据源和行列信息
    final String dsId = parser.getItem1();
    final int row = Integer.parseInt(parser.getItem2() );
    final int col = Integer.parseInt(parser.getItem3() );
    //获取对应数据源的记录集
    ArrayList <ArrayList <String >> rows = mResuleSet.get
(DS_PREFIX+dsId);

    if(rows == null) {
        return;
    }

    String val = "";
    //获取标签文本对应的列值
    if( (row < rows.size()) && (col < rows.get(0).size() ) ) {
        try {
            val = rows.get(row).get(col);
        } catch(IndexOutOfBoundsException e) {
            return;
        }
    }
```

```
        //替换标签文本
        mWordUtil.replace(parser.getRawText(), val);
}
```

3.3 Word 工具类

Word 工具类是操控 Word 文档的核心代码，限于篇幅，代码 8 中只列举其部分代码。

代码 8　Word 工具类

文件名：MsWordUtil.java

```
//连接应用程序服务
@Override
public boolean connect(boolean isVisible) {
        try {
                //连接 Word 应用程序
                mWordApp = new ActiveXComponent ("Word.
Application");
                //设置应用程序界面可见
        mWordApp.setProperty("Visible", new Variant(isVisible) );
                return (true);
        } catch(Exception e) {
                e.printStackTrace();
                FooDebug.getInstance().println("连接应用程序服务
异常，请确认正常安装 WORD 程序！ ");
        }
        return (false);
}
//以隐藏方式连接应用程序服务
@Override
public boolean connect() {
        return(connect(false) );
}

//断开与应用程序服务的连接
@Override
public boolean disconnect() {
        if(this.mWordApp == null) {
                return (true);
        }

        try {
                mWordApp.invoke("Quit", new Variant[] {} );
                mWordApp = null;
                return (true);
        } catch(Exception e) {
                e.printStackTrace();
                FooDebug.getInstance().println("退出应用程序异
常！ ");
        } finally {
                mWordApp = null;
        }
        return (false);
}
```

3.4 解析文档模板标签文本

代码 9 是代码 6 中解析文档模板中标签文本的关键代码。

代码 9　解析文档模板标签文本

文件名：TagParser.java

```
private void parse() {
        if(this.mRawText.indexOf(RS_TAG) ! = FLAG_NOT_
FOUND) { //[%=RS(16,5,1)%]
                mType = TAG_TYPE_RS;
                int pos1 = this.mRawText.indexOf('(');
                int pos2 = this.mRawText.indexOf(',', pos1+1);
                int pos3 = this.mRawText.indexOf(',', pos2+1);
                int pos4 = this.mRawText.indexOf(')', pos3+1);

                this.mItem1 = this.mRawText.substring (pos1+1,
pos2); //数据源标识
                this.mItem2 = this.mRawText.substring (pos2+1,
pos3); //行号
                this.mItem3 = this.mRawText.substring (pos3+1,
pos4); //列号
        }
        else {
                mLogger.log (Level.WARNING, "Invalid tag: " +
mRawText + "! ");
        }
}
```

代码 9 中，为解析结果包含 3 项内容，其第一项为数据源标识，第二项为行号，第三项为列号。该标签文本由开发者进行定义，可以对项的数量和内容进行扩充。

4 结语

在线生成 Word 文档实际上是办公需求（办公文档处理）与 B/S 应用（门户网站等）的结合，其既可实现 B/S 应用下的共享方式，又延续了处理办公文档的习惯。

在文中，一方面巧妙地使用了 Java 到 COM 组件的桥接库，方便地解决了在 Java 平台操纵 COM 组件的困难；另一方面，使用 Word COM 组件，实现了将 Word 文档的操控发挥得淋漓尽致（包括：全文检索和全文替换等），使得案例的实用性增色不少。

最后，需要指出的是，Word COM 组件只能运行于 Windows 系统，JACOB 对 COM 组件的使用也暂时局限于 Windows 系统，这无疑限制了 Web 应用跨平台的实现，这一点可能需要开发者根据项目的实际要求进行取舍。

（收稿日期：2012-08-15）

基于 ASP.NET 技术的企业信息网站设计与实现

李志云

摘　要：很多大中型的企业都通过自己的网站来吸引客户，扩展自己的业务市场。讲解了利用 ASP.NET+Access 设计和实现一个企业网站的过程和方法。

关键词：企业网站；ASP.NET 技术；C# 语言

1　引言

企业宣传是企业日常工作中的重要一环，通过宣传可以让社会各界都了解企业的基本情况，从而扩大企业在市场中的竞争力。目前，很多大中型的企业都通过自己的网站来吸引广大的客户，从而扩展自己的业务市场。下文给出了某发电设备公司网站的设计与实现过程。图 1 是该网站运行时的首页。

图 1　网站首页

2　开发环境

Visual Studio 2008+Access 2003

3　后台数据库结构

该网站后台数据库采用 Access，数据库名称:chenfengdb。存在于网站的 App_Data 目录中。数据库中包括 8 个数据表：公司简介表、产品种类表、产品表、问题解答表、留言表、资质证明图片表、联系方式表和后台管理用户表。图 2 到图 9 是这些表的结构。

（1）公司简介表（tbIntroduce）用于保存公司介绍的信息，其结构如图 2 所示。

tbIntroduce : 表

字段名称	数据类型
id	自动编号
titles	备注
contents	备注

图 2　公司简介

（2）产品种类表（tbProductType）：用于保存所有的产品类型信息，其结构如图 3 所示。

tbProductType : 表

字段名称	数据类型
id	自动编号
titles	文本
vxmarks	备注
contents	备注
paths	备注

图 3　产品种类

（3）产品表（tbProducts）：用于保存所有的产品信息，其结构如图 4 所示。

tbProducts : 表

字段名称	数据类型
id	自动编号
titles	文本
contents	备注
vxmarks	备注
paths	备注

图 4　产品信息

（4）问题解答表（tbQuestions）：用于保存所有的问题信息，其结构如图 5 所示。

tbQuestions : 表

字段名称	数据类型	说明
id	自动编号	
titles	文本	
contents	备注	

图 5　问题解答

（5）留言表（tbLeaveWords）：用于保存所有的留言和留言回复信息，其结构如图 6 所示。

图 6　留言信息

(6) 资质图片表 (tbNewspic)：用于保存所有的证件图片信息，其结构如图 7 所示。

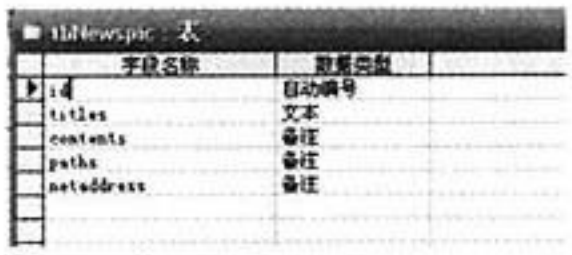


图 7　资质图片

(7) 联系方式表 (tbTel)：用于保存公司的联系方式信息，其结构如图 8 所示。

图 8　联系方式

(8) 后台管理用户表 (tbUser)：用于后台管理人员的信息，其结构如图 9 所示。

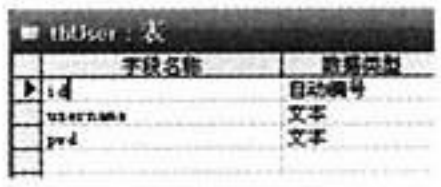


图 9　后台管理

4　网站的文件夹结构

网站的文件夹结构如图 10 所示。

图 10　网站结构

5　在配置文件中设置到后台数据库的连接

在网站配置文件 web.config 的<connectionStrings>节中添加如下代码：

```
<connectionStrings>
<add name="chenfengdbConnectionString" connectionString
="Provider=Microsoft.Jet.OLEDB.4.0;
Data Source=|DataDirectory|chenfengdb.mdb" providerName
="System.Data.OleDb"/>
</connectionStrings>
```

6　网站主页

主页是进入网站后首先显示的页面，通过主页上的超链接可以进入到其他页面。发电设备网站的主页包含导航栏、滚动图片、产品种类、产品展示和客户留言等内容。

网站的页面采用表格布局，页面中主要使用 DataList 数据控件来显示后台数据库中的产品种类、问题解答和产品展示等信息。其中用于显示产品列表的 DataList 控件的前台代码如下：

```
<asp:DataList ID = "DataList4" runat = "server"
RepeatColumns="4"
    style = "text –align: center" Width = "609px"
RepeatDirection="Horizontal">
  <ItemTemplate>
    <table align = "center" cellpadding = "0" cellspacing = "0"
style="width: 120px; height: 110px">
      <tr>
        <td class="tdd–center"> <a href='Products.aspx?
ids =<% # DataBinder.Eval (Container.DataItem,"id") % >'
><asp:Image ID = "Image1" runat = "server" Height = "90px"
ImageUrl='<%# DataBinder.Eval(Container.DataItem,"paths")
% >' Width = "110px" /></a><br /><a href ='Products.aspx?
ids =<% # DataBinder.Eval (Container.DataItem,"id") % >' >
<asp:Label ID="Label4" runat="server"Text='<%# GetNewStr
(Convert.ToString(DataBinder.Eval(Container.DataItem,"titles")),
10) %>'></asp:Label></a>
        </td>
      </tr>
    </table>
  </ItemTemplate>
</asp:DataList>
```

用于配合 DataList 控件的主页后台代码如下：

```
OleDbDataAdapter da4 = new OleDbDataAdapter ("
select top 24 titles,paths,id from tbProducts order by id
desc", con);
DataSet ds4 = new DataSet();
da4.Fill(ds4);
this.DataList4.DataSource = ds4;
this.DataList4.DataBind();
```

```
con.Close();
```

7 子页的母版页

网站中的其他页面基本都是依托母版页来实现的。图 11 是子页的母版页界面。

图 11 母版设计

8 后台管理登录页面

图 12 是后台用户登录时的页面。

图 12 后台登录页

其中登录按钮的代码如下：

```
protected void btnLogin_Click(object sender, EventArgs e)
{
    try
    {
        if (txtUsername.Text == '' || txtPwd.Text == '')
        {
            this.Page.RegisterStartupScript ("ss", " <script >
alert(' 用户名称和密码信息不能为空！ ')</script>");
            return;
        }
        else
        {
            string num = this.txtValidate.Text.Trim();
  if (Session["ValidNums"].ToString() == num.ToUpper())
            {
                // 连接数据库
                                OleDbConnection con = new
OleDbConnection (ConfigurationManager.ConnectionStrings["
chenfengdbConnectionString"].ConnectionString);
                con.Open();
                string strselect = "select * from tbUser where
username='" + txtUsername.Text.Trim() + "'";
                                OleDbDataAdapter das = new
OleDbDataAdapter(strselect, con);
                DataSet dss = new DataSet();
                das.Fill(dss);
                if (dss.Tables[0].Rows.Count > 0)
                {
                    string strselect1 = "select * from tbUser
where username ='" + txtUsername.Text.Trim () + "' and
pwd='" + txtPwd.Text.Trim() + "'";
        OleDbCommand sqlcmd1 = new OleDbCommand
(strselect1, con);
    OleDbDataReader dr1 = sqlcmd1.ExecuteReader();
                    if (dr1.Read())
                    {
                        //Request["userid"] = dr1["id"].ToString();
                Session["username"] = txtUsername.Text;
                        Response.Redirect("Manages.aspx");
                    }
                    else
                    {
                        this.Page.RegisterStartupScript ("ss", "<
script>alert(' 密码错误！ ')</script>");
                        return;
                    }
                    dr1.Close();
                }
                else
                {
                    this.Page.RegisterStartupScript ("ss", " <
script>alert(' 用户名不存在！ ')</script>");
                    return;
                }
                con.Close();// 关闭连接
            }
            else
            {
                this.Page.RegisterStartupScript ("ss", "<script>
alert(' 验证码输入错误！ ')</script>");
                return;
            }
        }
        catch (Exception ex)
        {
            this.Page.RegisterStartupScript ("ss", "<script>alert('
验证码输入错误,请刷新页面！ ')</script>");
        }
    }
```

9 后台管理主页面

后台管理页面依然采用母版页设计，图 13 是后台管理用户的主操作页面。

图 13 后台管理主页

其中“删除”按钮的代码如下：

```
OleDbConnection con = new OleDbConnection(Configuration
Manager.ConnectionStrings ["chenfengdbConnectionString"].
ConnectionString);
    con.Open();
    OleDbCommand sqlcom;
    string sqlstr = "delete from tbProductType where id=" +
Convert.ToInt32 (GridView2.DataKeys [e.RowIndex].Value.
ToString()) + "";
    sqlcom = new OleDbCommand(sqlstr, con);
    sqlcom.ExecuteNonQuery();
    con.Close();
    AddLoadDatas(2); //调用显示信息的方法
```

10 结语

在发电设备网站的开发中，许多页面使用了母版页，这样可以使页面风格统一，而且提高了制作效率。显示后台数据信息时主要使用了 DataList 控件，DataList 控件在显示数据时通过编辑模板来实现，数据的显示非常灵活。另外，在网站中也大量使用了 JavaScript 脚本、CSS 样式表等技术。企业宣传网站是网站开发中经常遇到的问题，如何完成此类网站的开发，以上给出了切实有效的解决方案。

参考文献

[1] 高宏，李俊民，等. ASP.NET 典型模块与项目实战大全. 北京：清华大学出版社，2012.

（收稿日期：2012-08-27）

UIPower 的 DirectUI 3D 界面引擎成功发布

纵观目前市场上的软件产品，无一例外地都呈现出平面（2D）的界面效果。其实我们所处的这个世界是三维（3D）的。有时候为了能描述一个设备或仪器需要采用大量的参数化的控件来呈现，还要借助一大堆的文档来解释各个参数的含义与用法。由于受到开发工具的限制，很难轻松实现 3D 的 UI 效果。在几年前，腾讯的一款概念版的 QQ 令无数网友兴奋不已，原因是其超酷的多角度翻转打破了传统的人机交互的模式。可惜的是，要开发出如此绚丽的 UI 则需要借助微软的 WPF 工具。而总所周知，WPF 是一款基于.net3.5 框架的重量级的系统，由于无法向下兼容各种软硬件，从而无法在短期内得以普及。

随着硬件设备的处理能力的不断提升，就连小小的手机都能够轻松玩转大型的 3D 游戏：赛车、射击游戏等。而对台式机来说，硬件配置更高，处理能力更强，在各种应用软件上更需要能体现软件企业的开发实力和强大的人机交互水平。对于工业控制的软件来说，如何将复杂的设备呈现在计算机屏幕上，并能动态及时地反应出设备的各项技术参数，则显得尤为重要。如果能将设备控制软件与设备各项参数都动态显示在一个软件 UI 中，那么对于软件使用者来说，这显得非常的重要。对于触摸型大屏系统来说，如果能做到 UI 模拟真实场景帮助用户完成各项操作任务，那也将是非常直观的。比如医院的导医系统、KTV 点歌系统等。目前，实现这样的 3D 效果基本上都是采用 Flash 技术，而 Flash 最大的坏处就是占用资源大，运行效率不高，与各种应用程序的对接比较复杂。所以，目前市场需要有一款完全支持 3D 界面技术的界面开发工具来满足各个软件开发企业更高要求的 UI 开发需求。

UIPower 组建团队专门研发了 3D 界面引擎。从技术核心上彻底改变了以往的 2D 坐标系统，每个控件元素都支持 3 个坐标值：X、Y、Z。将传统的平面窗口改变为三维可旋转的窗口，用户可以多视角来操作界面上的内容。即便是在界面斜放或侧面的时候，都能够用鼠标准确点击里面的控件。这个交互体验完全实现了真实世界的用户体验。性能上做到了非常低的 CPU 占用和内存占用。一改以往复杂三维界面占用大量 CPU 资源的做法，UIPower 充分发挥显卡硬件的性能，将所有与图形计算相关统统交由 GPU 去处理。经过长时间的测试和验证，目前的绝大部分的电脑的配置都能流畅地运行该 3D 界面引擎。

* 3D 界面引擎的运行要求：

1、CPU：酷睿双核

2、内存：512M 以上

3、显卡：支持 DirectX9.0C 或以上，支持 ShadeMode2.0 或以上

4、DirectX 运行时：MS DirectX Redist (June 2010 或以上)

5、操作系统：WindowsXP、Vista、Windows 7、Windows 8

基于位图的数字时间显示

穆宣社 李兆旺

摘 要：在VC++环境下，利用10个个位数字与3个分隔符号的位图图片，为应用程序添加数字时间显示的方法与步骤。

关键词：位图资源；数字时间

在软件编程实践中，有时需要在程序适当位置显示当前的天文时间或者变换后的假想时间，即需要在应用程序界面上显示类似LED液晶时钟，以起到实时提醒当前时间和烘托气氛的作用。笔者在开发一个文件收发系统过程中，利用组成数字时间的各个元素的位图图片表示时间的各个数字和分隔符号，实现了应用程序的数字时间时钟显示功能，方法灵活，效果实用。下面介绍实现时钟功能的方法步骤，包括时钟类的定义及其在具体工程中的应用。

1 建立显示数字式时间的工程

首先在DigitalTimeTestDlg工程中添加13个图片，分别为数字0~9以及“：—”（最后一个为空格）。然后编好正确的ID号顺序，如IDB_BMPDT_0、IDB_BMPDT_1、IDB_BMPDT_2、.....、IDB_BMPDT_9，分别对应数字、、、.....、，而IDB_BMPDT_10、IDB_BMPDT_11、IDB_BMPDT_12则分别对应：  。用大小为15×25像素位图图片表示时间的各个数字和分隔符号，在VC++环境下实现了在应用程序中现实数字时间的功能。

（1）新建一个基于对话框的DigitalTimeTest工程

添加一个CDigitalTime对象 m_DigitalTime; 然后创建数字时间：

```
int CDigitalTimeTestDlg::OnCreate (LPCREATESTRUCT lpCreateStruct)
 {
    if (! m_DigitalTime.Create ("",WS_CHILD|WS_VISIBLE, CRect(0,0,0,0),this,NULL)) { return -1;}
          m_DigitalTime.SetStyle (CLEDClock::XDC_SECOND);//设置时间样式,显示到秒
          return 0;
 }
```

（2）显示数字时钟到父窗体指定的位置

```
void CDigitalTimeTestDlg::OnSize (UINT nType, int cx, int cy)
 {
    CDialog::OnSize(nType, cx, cy);
    CRect  rc;
    if(m_DigitalTime.GetSafeHwnd() ! = NULL)
    {
          this->GetClientRect(&rc);
          rc.left  = rc.right - 350;rc.right  = rc.right - 5; rc.top= 24;rc.bottom = 58;
          m_DigitalTime.MoveWindow(&rc);} //移动数字时钟的位置
 }
```

（3）将对话框在前端显示

```
void CDigitalTimeTestDlg::OnTopshow()
{
          this->SetWindowPos(&wndTopMost,0,0,0,0, SWP_NOMOVE|SWP_NOSIZE);
}
```

基于对话框的工程中所显示的数字时间运行界面如图1所示。

图1 运行界面

对于基于SDI和MDI的工程，在工具条上的显示数字时间的时钟时，与上述方法类似。

2 数字时间类的定义（CDigitalTime）

（1）基于静态标签控件定义数字时间类

```
class CDigitalTime: public CStatic
{
   protected:
     virtual void PreSubclassWindow();//子类化窗口,作用启动时钟,初始化小位图变量
     afx_msg void OnTimer(UINT nIDEvent);//时钟,在此获取并显示系统当前时间
```

```
    afx_msg void OnPaint();//初始化时钟显示
  private:
    UINT m_w; UINTm_h;//显示位图的高和宽
    HBITMAP m_hBitmap[13];//小位图句柄数组,包括 0~9,
//以及冒号和减号分隔符和空白符号共 13 个图片。
  void Output(UINT digit, UINT pos);//输出小时钟位图
  };
```

(2) 调入上述的 13 个小位图图片

顺序是 0~9、冒号分隔符、减号分隔符和空白符号:

```
  CLEDClock::CLEDClock()
  {
    for(int i = 0; i< 13; i++) {
    HBITMAP temp = (HBITMAP)::CreateMappedBitmap
(AfxGetApp()->m_hInstance, i+IDB_BMPDT_0, 0, NULL, 0);
        m_hBitmap [i] = (HBITMAP)::CopyImage ( temp,
IMAGE_BITMAP, 0, 0, LR_COPYRETURNORG | LR_COPYD
ELETEORG);}
  }
```

(3) 举例显示数字时间

```
  void CDigitalTime::OnPaint()
  {
  CPaintDC dc(this); // device context for painting
  /* 进行初始化的显示----以下以 2012 年 07 月 28 日 17:
28:56 为例输出为:
  2012-07-28 17:28:56 ,共由 19 个小图片显示而成 *//
  CRect rectClient; GetClientRect(&rectClient); //获取客户
//区域大小
m_h = rectClient.Height(); m_w = rectClient.Width()/19;
//获取时钟显示宽及高
SYSTEMTIME time; ::GetLocalTime(&time);
//获取系统当前时间
int nTmp1,nTmp2,nTmp3,nTmp4; //用于显示的临时变量
nTmp1 = time.wYear / 1000; Output(nTmp1,0); //计算并显
//示年的第一位,如"2"
nTmp2 = (time.wYear % 1000) / 100; Output (nTmp2,1);
//计算并显示年的第二位,如"0"
nTmp3 = ((time.wYear % 1000) % 100) / 10;Output
(nTmp3,2); //计算并显示年第三位"1"
nTmp4 = (((time.wYear % 1000) % 100) % 10) % 10;
Output(nTmp4,3); //显示年第 4 位,如"2"
Output(11,4); //第 5 个位置显示减号分隔符"-",为从 0 起第
//11 个图片
nTmp1 = time.wMonth / 10; Output(nTmp1,5); //第 6 个位置
//显示月的第一位,如"0"
nTmp2 = time.wMonth % 10; Output(nTmp2,6);//第 7 个位置
//显示月的第二位,如"7"
Output(11,7); //第 8 个位置显示减号分隔符"-",为从 0 起第 11
//个图片
nTmp1 = time.wDay / 10; Output(nTmp1,8); //第 9 个位置显
//示日的第一位,如"2"
nTmp2 = time.wDay % 10;Output(nTmp2,9); //第 10 个位置
//显示日的第一位,如"8"
  Output(12,10); //第 11 个位置显示空白分隔符" ",为从 0 起第
//12 个图片
nTmp1 = time.wHour / 10; Output(nTmp1,11); //第 12 个位置
//显示时的第一位,如"1"
nTmp2 = time.wHour % 10; Output(nTmp2,12); //第 13 个位
//置显示时的第一位,如"7"
  Output(10,13); //第 14 个位置显示冒号分隔符":",为从 0 起第
//10 个图片
nTmp1 = time.wMinute / 10; Output(nTmp1,14); //第 9 个位
//置显示分的第一位,如"2"
nTmp2 = time.wMinute % 10; Output(nTmp2,15); //第 16 个
//位置显示分的第一位,如"8"
Output(10,16); //第 17 个位置显示冒号分隔符":",为从 0 起第
//10 个图片
nTmp1 = time.wSecond / 10; Output(nTmp1,17); //第 18 个位
//置显示秒的第一位,如"5"
nTmp2 = time.wSecond % 10; Output(nTmp2,18); //第 19 个
//位置显示秒的第一位,如"6"
  }
```

(4) 在时钟中输出日期时间

```
  void CDigitalTime::OnTimer(UINT nIDEvent)
  {
    if(nIDEvent == m_nTimer) {
        m_nCount++;
  //以下与 CLEDClock::OnPaint() 中的初始化的显示相同}
    CStatic::OnTimer(nIDEvent);
  }
```

(5) 在指定位置输出各个小图片

```
  void CDigitalTime::Output(UINT digit, UINT pos)
  {
    CClientDC dc(this);//绘制位图
    CDC dcMem; dcMem.CreateCompatibleDC(&dc);
        CBitmap* pBitmap = CBitmap::FromHandle (
m_hBitmap[digit] );
        CBitmap* pOldBitmap = dcMem.SelectObject
(pBitmap);
    dc.StretchBlt(m_w*pos, 0, m_w, m_h,
    &dcMem, 0, 0, m_bm.bmWidth, m_bm.bmHeight,
SRCCOPY);//按设定宽高输出小图片
    dcMem.SelectObject(pOldBitmap);
    dcMem.DeleteDC();
  }
```

当然,以上调入的位图资源和数字时间在用完后需要释放,不再赘述。

(收稿日期:2012-08-24)

人民币纸币号码的快速识别

陶 胜

摘 要：探讨人民币纸币号码的快速识别，首先要将采集到的人民币纸币图像进行去噪、二值化、细化、剔除毛刺、字符分割等预处理，然后针对人民币纸币号码只有数字和大写字母的特殊性，设计出一种基于字符结构特征和拓扑特征的数字和字母识别方法。最后应用 Matlab 编程实现了人民币纸币号码的快速识别。仿真实验表明，该识别方法具有识别率高、速度快、抗噪能力强等优点。该方法对车牌识别和汉字手写体识别的研究有一定的借鉴和指导作用。

关键词：人民币；识别；细化；毛刺

1 引言

纸币上的号码是纸币印刷数量的标识，由于具有不重复性，因此可以用来标识纸币的身份。纸币号码自动识别在纸币印刷过程有着十分重要的应用价值，在出钞时自动识别并记录下纸币的号码，使出入银行和国家的纸币号码都有记录，一旦出问题，可以做到有据可查[1]。在打击经济犯罪方面也可以提供有力的证据，比如说与从自动取款机或点钞机结合，在点钞过程中实时性地记录下纸币号码，将纸币号码识别数据送入计算机处理，与被抢劫号码数据库比较，一旦有相同号码出现，便可确认目前流通的钱币为被抢劫的钱币，从而采取有力措施限制其流通，并能给公安机关提供破案线索和确凿证据[2]。若发现同一面值的纸币的号码相同，那么可以断定至少有一张假币存在，有助于识别假币。

人民币纸币号码由大写字母和数字组成。以百元大钞为例，首先用数码相机采集得到样本图像，经过细化、剔除毛刺等预处理之后，再从中提取各字母和数字的拓扑结构特征和结构特征，通过特征分析得出大写字母和数字的识别算法，在此基础上应用 Matlab 软件编程实现。

2 基本原理

纸币号码识别的基本原理图如图 1 所示。

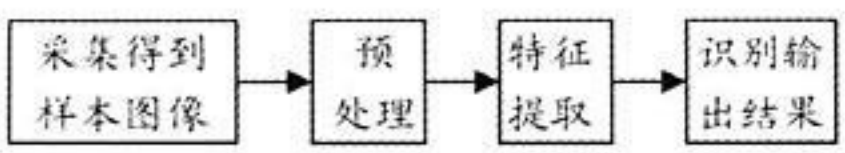


图 1 人民币纸币号码识别技术的基本原理图

百元大钞的大小是固定的，宽度为 15.6cm，高度为 7.7cm，因此用数码相机或 CCD 摄像机等设备采集得到的图像可以缩放到同一个大小，而号码固定在左下角区域，这样可以快速定位，从中提取号码样本图像，如图 2 所示。

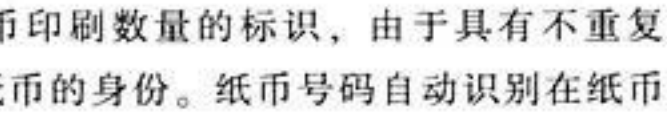


图 2 人民币纸币号码样本图像

该样本图像为真彩色图像，需要对其进行预处理。首先将真彩色数字图像转化为二值图像：

设 P 为真彩色数字图像的任意点，P 点颜色的 3 分量值分别为 r、g、b，则 P 的灰度值为 0.299*r + 0.587*g + 0.114*b，计算完各点的灰度值后，采用 Otsu 法确定一个全局阈值[3]，将各点的灰度值与该阈值比较，若灰度值大于等于阈值，二值化结果为 1，否则为 0。然后进行采用中值滤波方法对二值图像进行去噪处理。

接下来进行细化和去刺处理。图形在保持拓扑结构不变的条件下收缩为单点宽的结构称为细化[4]，也就是从原来的图中去掉一些点，但仍要保持原图的骨架[5]。那么怎样判断一个点是否能去掉呢？显然要根据它的 8 个相邻点的情况来判断，举例说明如图 3 所示。

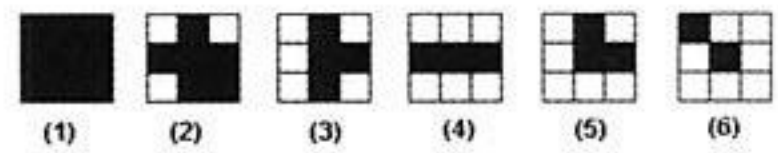


图 3 判断点能否被删除的示例

考查图 3 中的各中心点，(1) 不能删，因为它是个内部点，如果连内部点也删了，骨架也会被掏空的；(2) 不能删，和 (1) 是同样的道理；(3) 可以删，这样的点不是骨架；(4) 不能删，因为删掉后，原来相连的部分断开了；(5) 可以删，这样的点不是骨架；(6) 不能删，因为它是直线的端点，如果这样的点删了，那么最后整个直线也被删了，剩不下什么；另外，孤立点不能删，因为孤立点的骨架就是它自身。

由此得出判据：①内部点不能删除；②孤立点不能删除；

③直线端点不能删除；④如果P是边界点，去掉P后，若连通分量不增加，则P可以删除。

根据上述判据，事先做出一张表[3]，该表有256个元素，每个元素要么是0，要么是1。根据要处理的黑色点的8个相邻点的情况查表，若表中的元素是1，则表示该点可删，否则保留。

查表的方法是，设白点为1，黑点为0；左上方点对应一个8位二进制数的第一位（最低位），正上方点对应第二位，右上方点对应的第三位，左邻点对应第四位，右邻点对应第五位，左下方点对应第六位，正下方点对应第七位，右下方点对应的第八位，按这样组成的8位二进制数去查表即可。

实际细化过程中，删除点的顺序为：先是水平方向扫描第一行最左边的点；第一行最右边的点；第二行最左边的点；第二行最右边的点；……，最后一行最左边的点；最后一行最右边的点；然后是垂直方向扫描第二列最上边的点（因为第一列最上边的点已被删除）；第二列最下边的点；第三列最上边的点；第三列最下边的点；……，倒数第二列最上边的点（因为倒数第一列最上边的点已被删除）；倒数第二列最下边的点。

细化效果将直接影响识别速度及识别的准确率。经过细化处理后的图像可能会产生毛刺，为了得到正确的特征信息，必须将毛刺剔除。

由于样本图像为二值图像，只包括黑白两种颜色的像素，黑色像素称为前景图，代表实际字符信息，用“0”表示；将白色像素称为背景图，代表不包含字符信息的部分，用“1”表示。如果一个像素点本身是黑色，它的八领域像素有且仅有一个黑色像素点，则称该点为端点[4]。像素的连接数是指该点的8领域像素按逆时针顺序由0至1的跳变次数，显然端点的连接数为1，连接数为0的黑色像素点称为孤立点，连接数为2的黑色像素点称为连接点，连接数为3的黑色像素点称为分支点，连接数为4的黑色像素点称为交叉点，如图4所示。

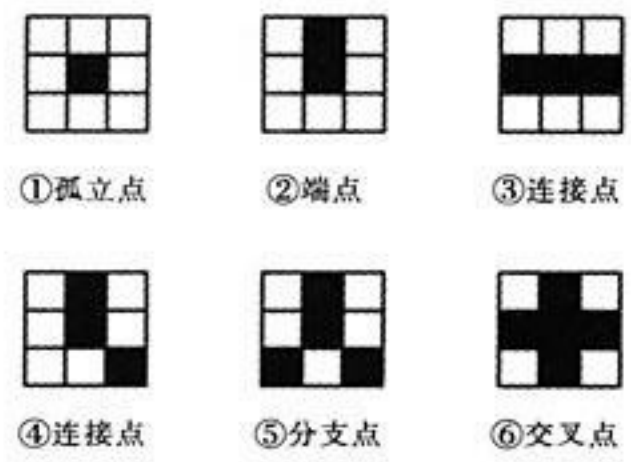


图4　连接数与点的类型

剔除毛刺的步骤如下：

步骤1：在全图范围内搜索端点，并记录下端点的坐标信息，由于端点的8领域像素只有一个黑色像素点，将该黑色相邻点置为当前点，将端点置为旧点，同时记录下当前点的坐标信息，置毛刺的初始长度n=1。

步骤2：考察当前点的8领域像素，分为以下几种情形：①这8个点中只有2个黑色点，其中之一为旧点，另一个点设为P点，将当前点置为旧点，P点成为当前点，记录下当前点的坐标信息，毛刺的长度n加1，转至步骤3；②这8个点中有3个黑色点，计算当前点的连接数，如果连接数为2，端点至当前点为毛刺，否则当前点为分支点，端点至当前点的旧点为毛刺，转至步骤4；③这8个点中有4个黑色点，即当前点为交叉点，端点至当前点的旧点为毛刺，转至步骤4；④这8个点中只有1个黑色点，当前点为端点，搜索完毕。

步骤3：判断毛刺的长度n是否超过给定的阈值N（N一般取4或5），若n小于等于N，转至步骤2；否则搜索完毕，不是毛刺。

步骤4：剔除搜索到的毛刺，即组成毛刺的各点像素值变为1。

步骤5：重复以上步骤，直到从所有端点出发搜索毛刺的过程完成。

图5~7显示了图2中的样本图像经过预处理后的效果。

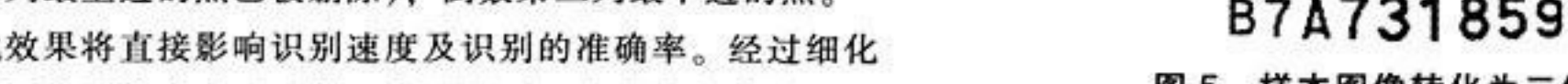


图5　样本图像转化为二值图像

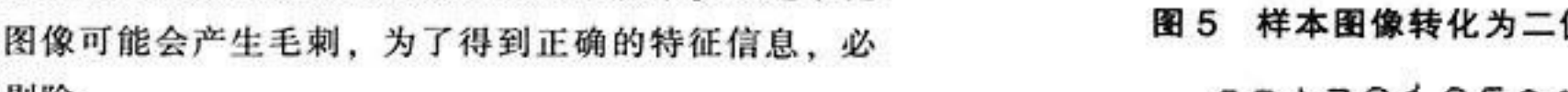


图6　细化后的图像

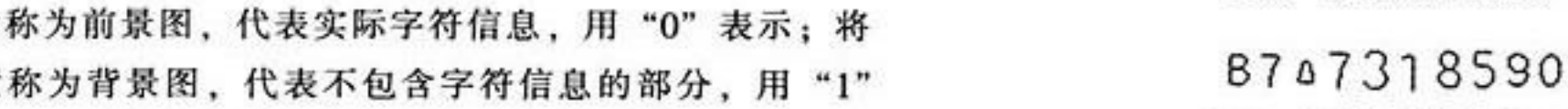


图7　剔除毛刺后的图像

完成预处理后，接下来就是提取每个大写字母和数字的特征。本文考虑拓扑结构和端点信息，作为分类识别的特征。

2.1　结构特征

不难发现各大写字母和数字的拓扑结构特征明显，字母“B”和数字“8”包含两个洞；字母“A”、“D”、“O”、“P”、“Q”、“R”和数字“0”、“4”、“6”、“9”含有一个洞；其他大写字母和数字均不含有洞。

2.2　端点特征

对26个大写字母进行分析，得出各字母的端点数和分支点数，如表1所示。

表1　大写字母的端点特征

字母	端点数	分支数	字母	端点数	分支数
A	2	2	N	2	0
B	0	2	O	0	0
C	2	0	P	1	1
D	0	0	Q	2	2
E	3	1	R	2	2
F	3	1	S	2	0

字母	端点数	分支数	字母	端点数	分支数
G	2	0	T	3	1
H	4	2	U	2	0
I	2	0	V	2	0
J	3	1	W	2	0
K	4	2	X	4	2
L	2	0	Y	3	1
M	2	0	Z	2	0

注：“A”和“Q”的两个端点与分支点的连线可能会被当成毛刺去掉，因此它们的端点信息从除刺前的图像提取。

对10个数字进行分析，得出各数字的端点数和分支点数，参见表2。

表2 数字的端点特征

数字	端点数	分支数	数字	端点数	分支数
0	0	0	5	2	0
1	2	0	6	1	1
2	2	0	7	2	0
3	2	0	8	0	2
4	2/1	1	9	1	1

注：“4”的端点与分支点的连线可能会被当成毛刺去掉，因此其端点数可能是2或1。

2.3 识别流程

根据提取的拓扑结构特征和端点特征，可以将26个大写字母和10个数字区别开来。先由欧拉数确定含有的洞数，进行区分，如“B”可以由欧拉数确定，不需其端点信息；若洞数相同，再由端点数和分支点数进一步区分，若端点信息一致，根据笔画信息、端点和分支点的位置及连接信息来确定各大写字母[6]。考察端点的周围8个点，如果上面的3个点中有1个黑点，则称该点为上有连接，如果上面的3个点全部为白点，则称该点为上无连接；同理，可定义下有连接、下无连接、左有连接、左无连接、右有连接、右无连接。因此提取端点特征，包括端点数和各个端点的坐标以及该端点的连接信息。

例如，“C”和“I”的端点数均为2，“C”的第二个端点上无连接，“I”的第二个端点上有连接；又如，“H”、“K”和“X”的端点数均为4，其中左右边都有竖线的为“H”；只有左边为竖线的为“K”；两边无竖线的为“X”。而端点数为3的字母“T”和“Y”，分叉点在上的为“T”；分叉点在中间的为“Y”。另外，字母“Q”被除刺后可能与“O”的信息一致，需要它的除刺前特征来识别。

大写字母和数字识别流程图如图8所示。

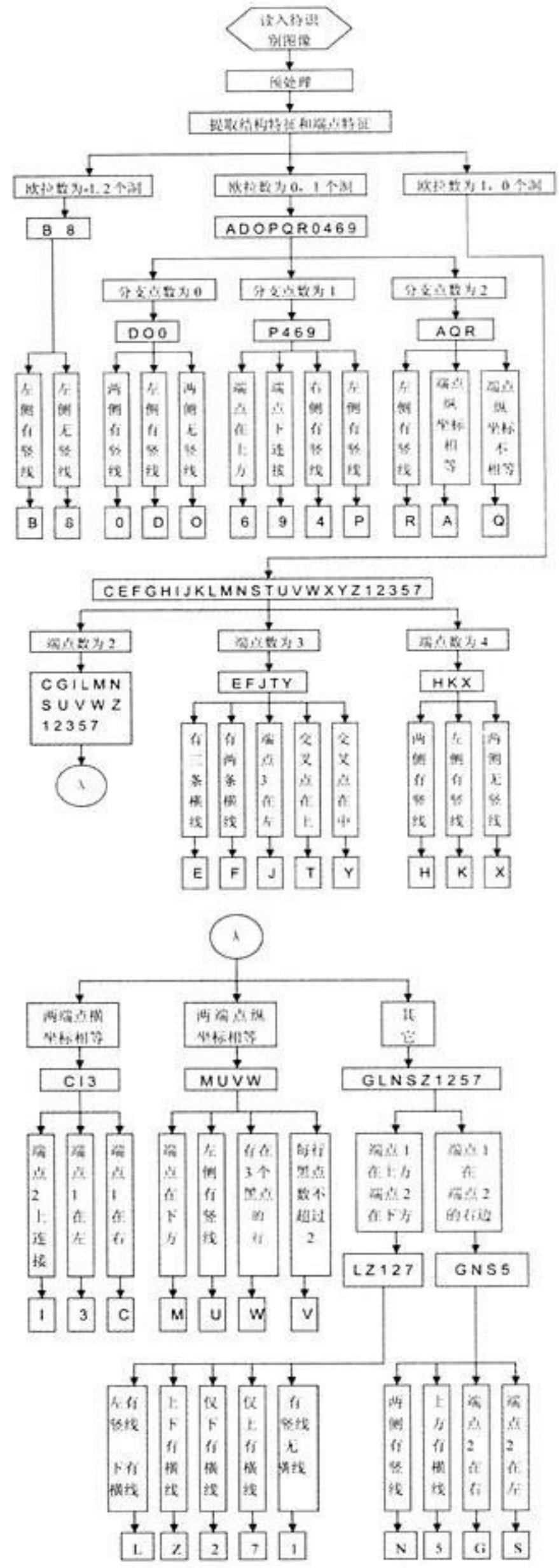


图8 识别流程图

3 实验结果

采用Matlab7作为开发工具，在Windows XP平台下实现

（下转第72页）

GUI 界面自动化测试的核心技术

仲光亮

摘　要：在 GUI 界面自动化测试的研发项目过程中，研究了核心技术键鼠的模拟操作，查证了目前资料存在各种问题，经研究和不断试验得到解决，并在实际的产品测试中得到了验证，并取得很好效果。

关键词：UNICODE 编码；扫描码；虚拟码；系统硬件输入队列；虚拟输入队列

1　GUI 界面自动化测试概述

在对软件进行的各种测试中，GUI 界面的功能测试是需人力时间等资源最多的。这是因为通常的软件界面功能测试要测试人员手工操作界面要素（如菜单、对话框等）来调用相关功能，输入测试数据，然后观察程序返回结果判断功能是否正确。

故欲实现 GUI 界面的自动化测试，须实现对其过程中的两个核心要素进行替代：一是使用计算机的自动输入来模拟替代测试人员的手工输入，具备测试输入自动化的特性；二是使用计算机对程序返回进行判断替代测试人员的人工判断，具备测试判断智能化的特性。下文对第一特性的核心技术进行全面分析研究，并系统完整地给出每一种技术的实现方式。

考虑到实际工作中多数测试在 Windows 平台上进行的，这里提及技术基于 Windows 平台，对于 UNIX 平台的 GUI 界面功能测试，则通过 Windows 上 X 程序如 X-WIN32,Xmanager,eXceed 等来进行。

2　Windows 操作系统的硬件输入原理

Windows 操作系统初始化时，会建立一个原始输入线程、一个系统硬件输入队列，操作系统接收从设备驱动程序中传来的硬件输入消息，放入系统硬件输入队列（简称 SHIQ）中，原始输入线程（简称 RIT）从 SHIQ 中取得硬件输入消息，解释后递送到目标线程。操作系统中其他工作线程（包括但不限于窗口线程）具有各自的虚拟输入队列（简称 VIQ）。RIT 通过不同的机制将鼠标和键盘输入递送到相应工作线程的 VIQ 中。Windows 操作系统通过该机制保证了各个进程的线程 VIQ 处于隔离状态，避免互相影响。

RIT 对鼠标输入递送机制是：RIT 首先通过调用 GetCursorPos 系统函数获得当前鼠标的光标位置，然后通过调用 WindowFromPoint 系统函数获得当前位置的窗口句柄，再通过调用 GetWindowThreadProcessId 系统函数获得当前窗口线程的 ID 信息，随后 RIT 将鼠标输入消息从 SHIQ 递送到该 ID 线程所在的 VIQ 中。

对于键盘输入采取的递送机制是：操作系统将只允许其窗口位于前端的线程和 RIT 进行对接，一旦该窗口被置后，操作系统将当前线程解开对接，将随后的前端窗口线程和 RIT 进行对接。在任一时刻，只允许一个线程与 RIT 进行对接。当一个键盘输入进入 SHIQ 后，RIT 取出并翻译成合适的按键消息，将其递送到对接线程的 VIQ 中。激活窗口的操作实际上及将创建该窗口线程和 RIT 对接的过程。对于建立多个窗口的线程，线程会将按键消息递送到拥有输入焦点的窗口（即 GetFocus 函数返回的窗口），确切地说是具有输入焦点的处于激活状态的前端窗口。

要将模拟的键鼠（指键盘鼠标，下同）输入递送到一个窗口，首先要将该窗口置为前端窗口，其次使其处于激活状态，最后使其具有输入焦点，这样模拟的键鼠输入才会进入正确的窗口。图 1 说明了输入从键鼠的硬件级别到窗口内的整个过程。

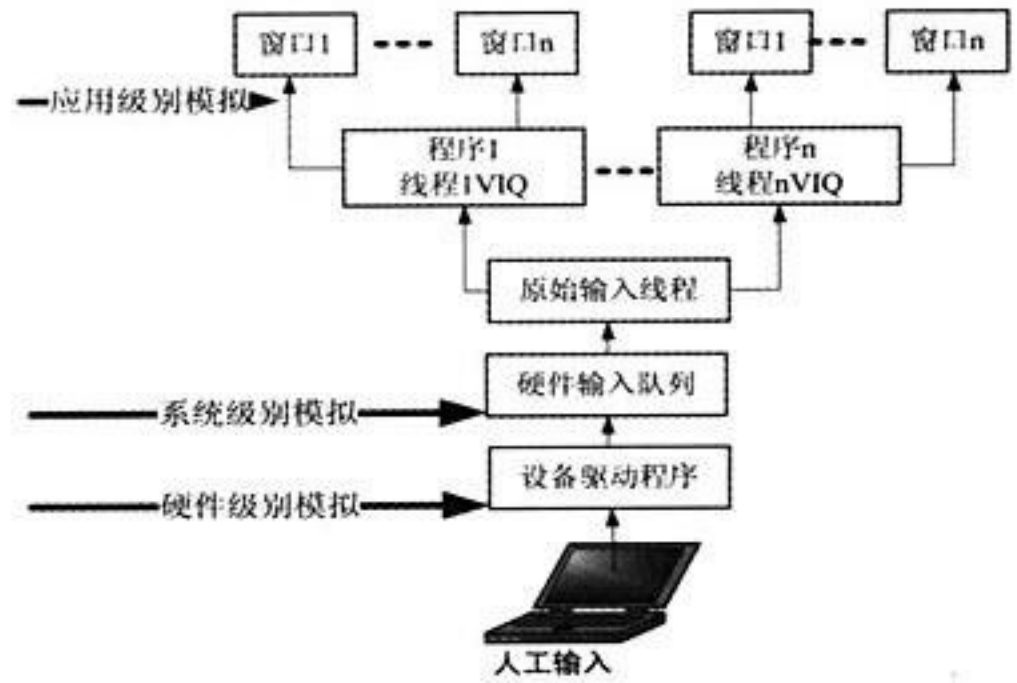


图 1　Windows 键鼠输入流程

以上流程分析可知有 3 个级别层次能实现用计算机来模拟人工输入到程序窗口，一是硬件级别的模拟端口输入。硬件级别的模拟操作多用设备驱动程序工作在 ring0 级别，这样破坏了系统完整性，影响系统运行稳定，此处不予讨论。二是系

统级别的模拟输入。三是应用程序级别的消息模拟。这两种方式都是利用系统提供的函数接口来实现，不存在第一方式的不足，接下来详细介绍这两种方式。

3 系统级别的模拟输入

系统级别的模拟输入是调用操作系统提供的函数，对键盘和鼠标的输入进行模拟操作，以达到替代人工、自动操作键鼠输入数据的目标。基本原理就是调用系统函数 SendInput、keybd_event、mouse_event 将封装好的键鼠事件放入 SHIQ 中，然后系统 RIT 取出放入 VIQ 中，最终输入到具有焦点的当前活动窗口。

这种方式的优点是可以模拟几乎所有的键鼠输入。缺点是：在进行系统级别的模拟输入操作时，被操作窗口必须是前端活动窗口，用户不能操作计算机，否则人手工产生的和计算机自动产生的键鼠输入容易导致混乱，因为二者都放入 SHIQ 中。

3.1 键盘的模拟操作

系统将键盘输入发送到拥有键盘焦点的前端窗口的消息队列中。键盘焦点是窗口的临时特性，系统中的所有窗口通过转移键盘焦点的方式来共享一个键盘，拥有键盘焦点的窗口将接收到所有的键盘输入。

键盘焦点是和活动窗口相关联的。活动窗口是指用户正在与之交互的最顶端可见窗口（高亮度标题条的窗口），活动窗口或者其子窗口拥有键盘焦点。可通过点击、ALT_TAB、任务管理器中选择的方式来设置当前活动窗口，应用程序也可调用系统函数来设置其创建的窗口为当前活动窗口。当一个窗口被设置为活动窗口时，系统将自动地把焦点设置到该窗口，随后所有的键盘输入将进入该窗口。

3.1.1 和键盘按键相关的编码

按键相关的编码分为扫描码、虚拟码和字符码。

其中扫描码是由键盘布局（硬件相关的扫描矩阵）决定的编码，对于给定的一种键盘其扫描码从应用软件编程角度来看是固定的，每一个键都有固定的扫描码，且按下和抬起都会产生扫描码。

虚拟码是由操作系统确定的与键盘硬件无关的一种编码，它和功能相关的，在应用程序中多使用虚拟码。操作系统根据键盘布局（系统安装时指定的键盘，记录在系统注册表中）的不同，在键盘驱动程序中将扫描码翻译成该种键盘布局对应的虚拟码。

字符码是和使用语言文字相关的一种编码，操作系统根据不同的语言文字将虚拟码映射到不同的字符编码上（如英语、德语、中文等），有许多种，比如 ASCII 码，GB2312 码，UNICODE 码等。

3.1.2 相关功能的详释

3.1.2.1 激活目标窗口成为顶端活动窗口的相关函数

要往目标窗口模拟发送键鼠信息，目标窗口必须成为顶端活动窗口，以使得模拟的键鼠信息进入目标窗口的 VIQ 中，即将窗口线程和 RIT 进行对接。核心代码如下：

```
/* 下面代码中的 hWnd 是要激活为顶端活动窗口的目标窗口，模拟发送的鼠标键盘消息将进入该窗口。*/
AllowSetForegroundWindow(-1); /* 允许调用进程设置顶端活动窗口 */
/* 获得目前的顶端活动窗口 */
HWND hForeWnd = GetForegroundWindow();
DWORD dwCurID = GetCurrentThreadId();
/* 获得目前顶端活动窗口的线程 ID*/
DWORD dwForeID = GetWindowThreadProcessId ( hForeWnd, NULL );
/* 将当前线程的输入处理机制附加到当前的顶端活动窗口线程上，以便能够正常地切换顶端活动窗口。*/
AttachThreadInput( dwCurID, dwForeID, TRUE);
if(IsIconic (hWnd))/* 如果最小化 */
    ShowWindow(hWnd,SW_RESTORE);/* 恢复原来的窗口 */
ShowWindow(hWnd,SW_SHOW);/* 显示窗口，强制刷新 */
SetForegroundWindow (hWnd);/* 设为顶端活动窗口，将该窗口的 VIQ 附加到 SHIQ 上，其后模拟发送的所有键鼠信息将会发送到该窗口。*/
AttachThreadInput( dwCurID, dwForeID, FALSE);
```

通过这样的方式来激活顶端窗口，可以不关注应用相关的输入焦点的复杂性，让程序主窗口处理其输入焦点问题。从而增强自动化测试脚本的通用性。

3.1.2.2 代码间转换的相关函数

程序可通过调用 MapVirtualKey 函数在虚拟码和扫描码之间进行转换，可通过调用 VkKeyScan、ToAscii、ToUnicode 来实现字符码和虚拟码之间的转换。

3.1.2.3 模拟输入相关的函数

两个系统函数 SendInput、keybd_event 能够实现键盘的模拟操作，它们都是由 user32.dll 动态库输出的。其中 keybd_event 一次调用只能发送一个击键事件，而 SendInput 一次调用可发送多个击键事件。在 Windows 操作系统的实现代码中，keybd_event 函数是调用内部函数 NtUserSendInput 来实现的，(SendInput 也是内部调用了 NtUserSendInput)，本文不再关注函数，(与此类似的有 mouse_event 函数)。SendInput 函数的原型：

```
UINT SendInput( UINT nInputs, /*input 事件的个数 */
LPINPUT pInputs, /* 欲插入的 input 事件数组 */
int cbSize/* input 结构的大小，注意：这里是一个 INPUT 结构的大小 */);
```

该函数的关键是构造 INPUT 结构，每个 INPUT 结构存放的是一个要插入 SHIQ 的键鼠事件。且一次调用可发送多个键鼠事件。该函数还有一很好特性，即其模拟发送的键鼠事件会连续地放入 SHIQ，不会被用户手工输入的或者其他 SendInput 发送的键鼠事件所打散。

INPUT 结构主体是联合，内有 3 个结构体，分别是

KEYBDINPUT、MOUSEINPUT、HARDWAREINPUT，可用来模拟鼠标、键盘、硬件3种不同的事件。会结合功能详解不同事件如何构造INPUT。

注意UIPI限制：该函数在Vista以后的操作系统上存在用户界面特权隔离（UIPI）的限制，一旦SendInput发送失败，仅从GetLastError的返回值中不能确定是功能失败还是被UIPI所阻止。故在Vista以后的操作系统运行时，需要注意程序的强制完整性控制（MIC）等级比目标程序高，后面的应用程序级别的模拟输入也要注意这个问题。

注意键盘状态：该函数不会重置当前键盘状态，如果SendInput函数调用时，当前正被按下的键将会干扰函数的正常执行。如果估计应用会出现这种状态，可通过GetAsyncKeyState来检查当前键盘状态来避免。

3.1.3 模拟键被按下和抬起

按键被按下和抬起是两个动作，为方便发送各种可能的组合按键，宜封装为两个函数，一个按下函数和一个抬起函数。

模拟按键被按下的代码如下：

```
/*isExtendKey是扩展键标志,扩展键包括101/102扩展键盘上的右ALT、右CTRL、右WIN、右MENU键,数字小键盘左面的插入、删除、起始、结束、上页、下页和上下左右4个箭头键,数字小键盘上的除法键、回车键。若要模拟发送以上按键需设置扩展键标志。*/
int siSendVK_KeyDown(unsigned char key,int isExtendKey)
    {
      INPUT input;
      UINT to_cnt = 1;
      UINT key_scancode = MapVirtualKey (key,0);/* 根据当前缺省的键盘布局,获得虚拟码对应的键盘扫描码。*/
      DWORD dwFlags = 0;
     /* 模拟发送的按键动作是KEYBDINPUT结构成员值来决定的。*/
input.type = INPUT_KEYBOARD; /* 模拟发送的是键盘信息 */
      input.ki.wVk = key;  /* 按键的虚拟码 */
    /* wScan成员很重要，一般情况其设为0不影响按键,但有些检查严格的程序不会接收wScan为0的按键信息。故其应设为wVk对应的扫描码,可通过MapVirtualKey获取虚拟码对应的扫描码,这样一般程序都会响应模拟发送的按键了。*/
      input.ki.wScan = key_scancode;
    /* dwFlags是设置不同按键情况的位标志，是扩展键要设置KEYEVENTF_EXTENDEDKEY位。如果是模拟按键的抬起操作,要设置KEYEVENTF_KEYUP位。*/
      if(isExtendKey ==1 )
          dwFlags = KEYEVENTF_EXTENDEDKEY;
      input.ki.dwFlags = dwFlags;
    /* dwExtraInfo成员应设置为当前线程消息队列的附加信息，通过GetMessageExtraInfo来获取该值。线程可以通过SetMessageExtraInfo来设置线程消息队列的附加信息。*/
      DWORD dwExtraInfo = GetMessageExtraInfo();
      input.ki.dwExtraInfo = dwExtraInfo;
    /*time成员是以毫秒为单位的时间戳,一般设置为0,系统将会自动提供内部时间戳。*/
      input[0].ki.time = 0;
      UINT sret = SendInput(to_cnt, &input, sizeof(INPUT));
      if(to_cnt ! = sret) {
          DWORD gle = GetLastError();/* 注意:判断返回值的时候,要考虑UIPI的影响因素,此处仅显示*/
          printf ("SendInput to_send=%d, send_ok =%d, error=%d\n",to_cnt,sret,gle);
          return 0;
      }
      return 1;
    }
```

模拟键抬起的函数int siSendVK_KeyUp (unsigned char key, int isExtendKey)，代码同siSendVK_KeyDown类似，就是dwFlags要设置KEYEVENTF_KEYUP标志，即：

```
input[0].ki.dwFlags = dwFlags|KEYEVENTF_KEYUP;
```

通过这两个函数的灵活组合调用，将能模拟发送任意可能的按键，包括组合键。

3.1.4 模拟发送单字节英文字符

对于可见的单字节ASCII字符，可通过调用VkKeyScan，将它转换为虚拟码，然后发送。发送序列是先模拟发送该ASCII字符的按下动作（siSendVK_KeyDown），然后再模拟发送一个抬起（siSendVK_KeyUp）动作。

注意大小写：大小写字母的发送，要判定大写锁定键是否开启（LED灯点亮），来决定是否模拟发送VK_SHIFT键序列，若发送其序列是先模拟发VK_SHIFT的按下，然后模拟发送字符键，最后模拟发送VK_SHIFT的抬起：

```
int siSendChar(char ch)
{
 int is_shift = (ch & 0x20); /* 判断大小写字母,如果是大写字母(ch & 0x20)将等于零,小写字母非零 */
 BYTE keysts[256];
 /* 判定大写锁定键是否开启（点亮），要注意不能使用GetAsyncKeyState(VK_SHIFT)来判断,因为该函数返回的是最近一次调月大写锁定键是否按下的状态，而不是大写锁定键的开启状态,GetAsyncKeyState只会返回一次正确状态，许多文章在这里存在错误,应使用GetKeyboardState系统函数。*/
 GetKeyboardState (keysts);
 BYTE cap_s = keysts[VK_CAPITAL];
 if ( cap_s & 0x1)
     is_shift = (is_shift == 0 ? 1 : 0);
 if(is_shift ==0) /* 需要做特殊处理,模拟SHIFT键按下,以便模拟发送大小写正确的字母 */
     siSendVK_KeyDown(VK_SHIFT,0);
 siSendVK_Key( VkKeyScan(ch));/* 获得该字符对应的虚拟码,并发送 */
 if(is_shift ==0)
     siSendVK_KeyUp(VK_SHIFT,0);/* 抬起SHIFT键 */
```

```
    return 0;
  }
```

对于英文字符串，一个一个字符地按照以上顺次处理发送即可。

3.1.5 模拟发送中英文混合字符串

在实际应用中，需要模拟发送中英文混合的字符串，其关键是先转换为 UTF-16 编码形式的宽字符串，然后还要判断转换完的字符是中文还是英文，再分别发送。

中文字的发送，首先 INPUT 中的 KEYBDINPUT 结构的 dwFlags 成员要设置 KEYEVENTF_UNICODE 标志位，然后将 wVK 成员设置为零，wScan 成员设置为中文的 UTF-16 格式的 UNICODE 编码：

```
void wsiSendUnicode(wchar_t data)
{
 INPUT input[2];
 input[0].type = INPUT_KEYBOARD;
 input[0].ki.wVk = 0;
 input[0].ki.wScan = data;//中文字符的 UNICODE 编码
    input[0].ki.dwFlags = KEYEVENTF_UNICODE;//发送
//UNICODE 编码
 input[0].ki.dwExtraInfo = GetMessageExtraInfo();
 input[0].ki.time=0;
 input[1].type = INPUT_KEYBOARD;
 input[1].ki.wVk = 0;
 input[1].ki.wScan = data;
   input [1].ki.dwFlags = KEYEVENTF_KEYUP |
KEYEVENTF_UNICODE;//按键抬起
 input[1].ki.dwExtraInfo = GetMessageExtraInfo();
 input[1].ki.time=0;
 SendInput(2, input, sizeof(INPUT));
}
```

中英文混合字符串的模拟发送代码示意如下：

```
void wsiSendKeys(CString msg)
{
/* 将中英文混合的字符串（中文是双字节的，英文是单字
节),转换成都是双字节的 UTF-16 编码形式的宽字符串,以便
统一处理。*/
 wchar_t* data = msg.AllocSysString();
 int i,len = wcslen(data);/* 取得宽字符串的长度 */
 SHORT vk;
for( i=0;i<len;i++){
    /* 如果宽字符小于 256,则表明是 ASCII 字符串 */
 if (data[i]>=0 && data[i]<256){
      vk = VkKeyScanW(data[i]);
      if (vk == -1)
                /* 如果取不到虚拟码,则认为是 UTF-16
格式的 UNICODE 字符 */
                wsiSendUnicode(data[i]);
      else
                siSendChar(data[i] &0xff); /* 能够取得扫描
码的,认为是单字节 ASCII 字符 */
    }
 else /* 是 unicode 字符,调用 wsiSendUnicode 函数发送 */
      wsiSendUnicode(data[i]);
    }
   ::SysFreeString(data);
}
```

3.1.6 模拟发送组合键

对于多个按键的组合键，其发送序列是按照每个键先按下后抬起的顺序依次发送。例如：CTRL+SHIFT+A 这个 3 键组合键的发送序列，代码如下：

```
siSendVK_KeyDown(VK_CTRL,0);
siSendVK_KeyDown(VK_SHIFT,0);
siSendVK_KeyDown('A',0);
siSendVK_KeyUp('A',0);
siSendVK_KeyUp(VK_SHIFT,0);
siSendVK_KeyUp(VK_CTRL,0);
```

这样就会向顶端活动窗口模拟发送一个 CTRL+SHIFT+A 的组合键。

3.2 鼠标的模拟操作

鼠标是操作系统所有窗口共享的。先封装一个鼠标模拟操作的函数，将填写 INPUT 中的 MOUSEINPUT 结构成员的代码予以封装，以方便后面的功能讲解。

```
 int siSendMouse (DWORD dwFlags,DWORD dx,
DWORD dy,DWORD dwData,DWORD dwExtraInfo)
 {
  INPUT input;
  UINT to_cnt = 1;
  input.type = INPUT_MOUSE;
  input.mi.dx = dx;
  input.mi.dy = dy;
  input.mi.mouseData = dwData;
  input.mi.dwFlags = dwFlags;
  input.mi.time = 0;
  if(dwExtraInfo == 0)
      dwExtraInfo = GetMessageExtraInfo();
  input.ki.dwExtraInfo = dwExtraInfo;
  UINT sret = SendInput(to_cnt, &input, sizeof(INPUT));
  if(to_cnt ! = sret) {
      DWORD gle = GetLastError();
      if(ShowRunInfo(SHOW_ERROR_INFO))
          printf("SendInput send_cnt=%d, send_ok =%
d,error=%d\n",to_cnt,sret,gle);
  }
  return 0;
 }
```

3.2.1 模拟鼠标定位、移动操作

当移动鼠标时，操作系统会同步在屏幕上移动鼠标光标，

鼠标光标中有一个称为作用点的像素，操作系统跟踪并将该作用点识别为鼠标的位置。

鼠标定位的模拟操作较简单，直接调用系统函数SetCursorPos (x,y)，就能将鼠标光标位置定位到相对于屏幕左上角的x（水平方向），y（垂直方向）位置。

鼠标移动的模拟操作：是指鼠标相对于当前位置移动，dx，dy指定移动的距离，移动方向由位置数据的正负来决定。X的正方向是从屏幕由左到右，Y的正方向从屏幕由上到下。另外移动距离与屏幕上鼠标光标的像素位置要进行换算，换算方法是将屏幕上像素距离映射到0~65535之间：

```
/*x,y 分别是要移动的屏幕水平和垂直方向的像素数 *，scnx,scny 是指当前屏幕分辨率的宽和高/
    POINT pt;
    /* 因为移动都是相对当前位置，所以要取得当前鼠标的光标位置,单位:屏幕像素 */
    GetCursorPos(&pt);
    /* 屏幕像素数和鼠标移动距离的换算公式:(要移动的像素距离 + 当前位置 )*65536/( 屏幕宽度或高度 -1)。MOUSEEVENTF_MOVE|MOUSEEVENTF_ABSOLUTE 标志表明模拟发送鼠标移动操作。*/
                    siSendMouse (MOUSEEVENTF_MOVE|MOUSEEVENTF_ABSOLUTE,
  (x+pt.x)*65536/(scnx-1), (y+pt.y)*65536/(scny-1),0,0);
```

3.2.2 模拟鼠标点击操作

鼠标有左、中、右3个键，动作有按下、抬起、单击、双击打、拖动5个动作，这样就有多种组合，所有动作都是在鼠标光标的当前屏幕位置进行。下面以鼠标左键为例，其余键相同。

模拟发送鼠标左键按下动作：

```
siSendMouse (MOUSEEVENTF_LEFTDOWN,0,0,0,0);
```

模拟发送鼠标左键抬起动作：

```
siSendMouse (MOUSEEVENTF_LEFTUP,0,0,0,0);
```

模拟发送鼠标左键单击动作，即连续发送按下和抬起动作：

```
siSendMouse (MOUSEEVENTF_LEFTDOWN,0,0,0,0);
siSendMouse (MOUSEEVENTF_LEFTUP,0,0,0,0);
```

模拟发送鼠标左键双击动作，连续发送2次按下和抬起动作即可：

```
siSendMouse (MOUSEEVENTF_LEFTDOWN,0,0,0,0);
siSendMouse (MOUSEEVENTF_LEFTUP,0,0,0,0);
siSendMouse (MOUSEEVENTF_LEFTDOWN,0,0,0,0);
siSendMouse (MOUSEEVENTF_LEFTUP,0,0,0,0);
```

模拟发送鼠标的拖动动作，先模拟发送按下动作，然后模拟发送鼠标移动，最后模拟发送鼠标抬起动作。其他各键请参照左键情况。

注意系统配置：一些系统配置参数会影响鼠标操作的动作行为，包括鼠标双击时间、鼠标左右键交换、鼠标滚轮滚动的行数等，应用程序可以通过SystemParametersInfo来获取或设置这些系统配置参数。

3.2.3 模拟鼠标滚轮操作

需要注意的是鼠标滚轮的滚动次数需要做系数调整，即乘以一个WHEEL_DELTA (=120)，代码如下：

```
int wv = atoi(argv[3])*WHEEL_DELTA;
siSendMouse (MOUSEEVENTF_WHEEL,0,0,wv,0);
```

这样就会使鼠标的滚轮前后滚动一次，滚动的方向有滚动数值的正负决定，负数向后滚动，正数向前滚动。

4 应用程序级别的模拟输入

应用程序级别的模拟输入原理是基于Windows操作系统的消息机制，通过向目标窗口发送键鼠消息，来实现输入的模拟操作。

这种方式的优点：可实现异步后台发送模拟操作，目标窗口不必是前端活动窗口，用户可在该电脑上从事其他工作，不会互相影响。缺点一：因目标程序实现机制的差异导致其响应处理的消息集不同，使得不能模拟所有的键鼠输入，例如Xmanager程序就只接受WM_KEYUP和WM_KEYDOWN两个键盘输入消息。缺点二：向目标窗口发送的消息不会改变键盘的状态。比如SHIFT状态，即使向目标窗口发送了正确的SHIFT键按下消息，在程序中调用GetKeyState或GetAsyncKeyState函数得到的还是SHIF没有按下T的状态。

和发送消息相关的系统函数有SendMessage，PostMessage，SendMessageTimeout等，通常情况下使用前两个，第一个函数需要等待消息处理完毕后返回，第二个不等待消息处理结果立即返回，视情况选用，文中使用前两个。

注意避免死锁：因SendMessage系统函数的同步特性，使用这个函数时要注意消息死锁导致的应用程序永久死锁，如果在测试中出现这种情况，可以考虑使用SendMessageTimeout函数(带有超时特性)，或者在被测程序处理中通过IsSendMessage系统函数判断是否为其他线程发送的消息，从而决定是否处理。

注意编码格式：存在两种编码形式的窗口，一种是采用UNICODE编码窗口，一种是采用ANSI编码窗口。其处理消息的窗口函数不同，UNICODE编码窗口消息处理函数DefWindowProcW，而ANSI编码的是窗口消息处理函数DefWindowProcA。不同编码格式的窗口对输入消息的处理有差异，可通过IsWindowUnicode系统函数来判断目标窗口是否为UNICODE编码窗口。

在向目标窗口发送文字时，涉及到文字的编码格式，通常情况下安装的Windows操作系统采用本地编码，中文Windows采用代码页为WINDOWS-936的字符集编码，即GBK编码，GBK编码与GB2312编码兼容的，GBK是GB2312的超集。另外一种Windows操作系统常用的编码是UTF-16形式的UNICODE编码。LINUX操作系统通常采用UTF-8编码。乱码

问题多由编码格式不一致造成，不同的编码格式之间可以转换。

4.1 模拟键盘输入的相关消息

本项目用到的和模拟键盘输入的相关消息有 WM_KEYDOWN、WM_SYSKEYDOWN、WM_KEYUP、WM_SYSKEYUP 4 个按键消息。WM_CHAR、WM_IME_CHAR 两个字符消息，字符消息和编码格式相关，不同编码格式的相同字符，其消息参数是不同的。

按键消息基本原理是，键盘被按下或抬起，系统 RIT 会产生 WM_KEYDOWN、WM_SYSKEYDOWN、WM_KEYUP、WM_SYSKEYUP，这些消息最终会放入拥有键盘焦点窗口的消息队列中，该窗口的窗口处理函数（两种编码形式）会响应并处理消息，从而实现键盘的模拟输入。按键消息的 wParam 参数是按键的虚拟码，lParam 参数包括重复次数、扫描码和各种标志。WM_SYSKEYDOWN 和 WM_SYSKEYUP 是系统按键（ALT 的组合键）产生的，WM_KEYDOWN 和 WM_KEYUP 是非系统按键产生的，程序根据需要来决定响应哪些消息。

字符消息 WM_CHAR（WM_SYSKEY 对应的是WM_SYSCHAR）是由 TranslateMessage 系统函数接收到 WM_KEYDOWN（或 WM_SYSKEYDOWN）消息后转换的（并根据 SHIFT 键和大小写锁定的状态来调整大小写）。所有字符消息的 wParam 参数包含了字符的编码，lParam 参数同按键消息。

WM_CHAR 的消息序列是（以字母 A 为例）：先收到按键‘A’的 WM_KEYDOWN，然后是字符‘a’的 WM_CHAR，最后是按键‘A’的 WM_KEYUP。

WM_SYSCHAR 多应用在菜单快捷键上，其消息序列是(以 ALT+A 为例)：先是 ALT 按键的 WM_SYSKEYDOWN，然后是按键‘A’的 WM_SYSKEYDOWN，接着是字符‘a’的 WM_SYSCHAR，再接着是按键‘A’的 WM_SYSKEYUP，最后是 ALT 按键的 WM_KEYUP（注意：最后是 WM_KEYUP，而非想当然的 WM_SYSKEYUP）。

WM_IME_CHAR 消息是由 IME（输入法程序）向目标窗口发送的，其消息参数中含有转换后的字符编码信息。

4.1.1 向 UNICODE 编码的窗口模拟发送中英文字符串

对采用 UNICODE 编码的窗口（使用 RegisterClassW 函数注册窗口类）来说，向其发送 WM_CHAR 和 WM_IME_CHAR 效果是一样，这两个消息是等同的，其消息的 wParam 参数含有字符的 UNICODE 编码：

```
/* 向 UNICODE 编码目标窗口 hw 发送 UNICODE 编码的字符 ch,postmsg 非零使用 PostMessageW 函数 否则使用 SendMessageW 函数，带 W 尾字母表明是 UNICODE 版本的系统函数 */
int spmwSendChar (HWND hw, UINT msg, WORD ch,int postmsg)
{
    MySendWndMsg pswm = SendMessageW;
    if(postmsg)
        pswm = (MySendWndMsg )PostMessageW;
    /* 将本线程的输入机制附加到目标窗口所属的线程上 */
                DWORD  TargetProcID =0,TargetThdID  =
GetWindowThreadProcessId(hw, &TargetProcID);
    DWORD MyThreadID =GetCurrentThreadId();
    AttachThreadInput(MyThreadID, TargetThdID, TRUE);
    pswm(hw,msg,ch,1);
    AttachThreadInput(MyThreadID, TargetThdID, FALSE);
    return 1;
}
/* 发送 UNICODE 字符串 */
int spmcwSendStr(HWND hw, wchar_t *sndstr,int postmsg)
{
    int i;
/* wcslen 取得 UNICODE 字符串的长度 */
    if(sndstr == NULL || wcslen(sndstr) <=0 )
        return 0;
    for(i=0;i<wcslen(sndstr);i++)
    {
    /* 中英文字符的 UNICODE 编码长度都是 2 个字节,其他语言的字符不一定。*/
        WORD ch = sndstr[i] ;
    /* 调用 WM_CHAR 消息递送函数发送 UNICODE 字符，这里也可以使用 WM_IME_CHAR 消息，两个消息对于 UNICODE 编码的窗口完全等同。将字符的 UNICODE 编码放入 wParam,lParam 参数置为 1。*/
        spmwSendChar (hw,WM_CHAR,ch,postmsg);
    }
    return 1;
}
```

4.1.2 向 ANSI 编码的窗口模拟发送中英文字符串

使用 ANSI 编码（使用 RegisterClassA 函数注册窗口类）格式的窗口，其若不支持 UNICODE 编码，这时向被测窗口模拟发送中英文字符串，就不能使用 UNICODE 编码，而必须使用和 ANSI 编码体系兼容的 MBCS（多字节）编码，Windows 操作系统多是 DBCS（双字节）编码，目前中文 Windows 都是和国标 GB2312 兼容的 GBK 编码。在此情况下 WM_CHAR 和 WM_IME_CHAR 消息是不一样的，其区别在消息的 wParam 参数的高低字节上：

```
/* 向 ANSI 编码目标窗口 hw 发送 DBCS 编码的字符 ch,postmsg 非零使用 PostMessageA 函数 否则使用 SendMessageA 函数，带 A 尾字母表明是 ANSI 版本的系统函数 */
int spmaSendChar (HWND hw, UINT msg, WORD ch,int postmsg)
{
    MySendWndMsg pswm = SendMessageA;
    if(postmsg)
        pswm = (MySendWndMsg )PostMessageA;
    /* 将本线程的输入机制附加到目标窗口所属的线程上 */
```

```
                DWORD TargetProcID=0,TargetThdID =
GetWindowThreadProcessId(hw, &TargetProcID);
    DWORD MyThreadID =GetCurrentThreadId();
    AttachThreadInput(MyThreadID, TargetThdID, TRUE);
    pswm(hw,msg,ch,1);
    AttachThreadInput(MyThreadID, TargetThdID, FALSE);
    return 1;
}
```

使用 WM_CHAR 消息模拟发送中英文字符串：

```
int spmcaSendStr (HWND hw, const char *sndstr,int
postmsg)
{/* 对于 ANSI 编码的字符串,判定长度使用 lstrlen*/
 if(sndstr == NULL || lstrlen(sndstr) <=0 )
      return 0;
 for(int i=0;i<lstrlen(sndstr);)
 {
      WORD ch = sndstr[i] ;
/* 由于对非 UNICODE 编码的 ANSI 窗口而言，输入的字
符可以单字节字符(如可见 ASCII 字符),也可以是双字节字符
(如果中文 GBK 字符)，不同性质的字符，其 WM_CHAR 或者
WM_IME_CHAR 消息的 wParam 是不同的，系统函数
IsDBCSLeadByte 就是对每一个字符进行判断,如果返回真,就
表明该字符是 DBCS 编码的前导字节，要和其后紧随一个字符
来共同表示一个双字节字符。
【注意】IsDBCSLeadByte 使用的是 ANSI 代码页,如果使用
其他代码页来判定 DBCS 编码使用 IsDBCSLeadByteEx 系统
函数。*/
      if(IsDBCSLeadByte(sndstr[i] ))
      {
/*【注意字节序】
要发送 DBCS 编码的双字节字符,第一个字节是前导字节,
以表明该字符是否是 DBCS 字符,后一个字符是延续字节。
若用 WM_CHAR 消息来模拟发送，其消息的 wParam 参
数低 8 位是前导字节,高 8 位是延续字节。
若用 WM_IME_CHAR 消息来模拟，其消息的 wParam 参
数高 8 位是前导字节,低 8 位是延续字节。
两个字符消息的 wParam 参数的高低字节含义正好相反。
使用 WM_IME_CHAR 消息的代码片段：
 ch =MAKEWORD (sndstr[i+1] ,sndstr[i] ) ;
 spmaSendChar (hw, WM_IME_CHAR,ch,postmsg);
 */
 ch =MAKEWORD (sndstr[i] ,sndstr[i+1] ) ;/* 注意字节序 */
          spmaSendChar (hw,WM_CAHR,ch,postmsg);
/* 使用 WM_CHAR 消息 */
          i = i +2;
      }
      else
      {/* 对以单字节的 ASCII 字符,WM_IME_CHAR 和
WM_CHAR 是一样的，这时因为 IME 输入法程序也可以输入
单字节的 ASCII 字符。*/
       spmaSendChar (hw, WM_CAHR,ch,postmsg);
          i = i +1;
      }
 }
 return 1;
}
```

4.2 鼠标消息

常用的窗口鼠标消息有鼠标移动消息 WM_MOUSEMOVE，以及鼠标左键的相关消息 WM_LBUTTONDOWN，WM_LBUTTONUP，WM_LBUTTONDBLCLK，其他鼠标中键右键类似。

消息的 wParam 参数表明 CTRL 键、SHIFT 键、鼠标左中右键是否按下。

消息的 lParam 参数表示鼠标光标的 X，Y 位置，坐标体系是相对于客户区域的左上角，低 16 位是 X 位置，高 16 位是 Y 位置。下面以鼠标左键单击为例，其他类似：

```
/* 在xpos,ypos 位置 单击 (包含按下和抬起两个动作)鼠
标左键 */
    SendMessage (targetHwnd,WM_LBUTTONDOWN,
MK_LBUTTON, MAKELPARAM(xpos,ypos));
    SendMessage (targetHwnd,WM_LBUTTONUP, 0,
MAKELPARAM(xpos,ypos));
```

注意：由于窗口鼠标消息是直接发往特定窗口的，可以后台进行，也不会产生真正的鼠标移动或者单击双击现象。只会触发相应鼠标消息对应的窗口处理过程，从而执行应用程序的相应功能。

5 其他高级特性

5.1 区分输入事件

有些应用场景要求能够区分键鼠事件的来源，即要分清楚事件是真正的键盘和鼠标硬件产生的，还是脚本程序模拟产生的，然后根据不同的来源来做相应的处理。实际测试中为避免人工输入干扰自动化测试的模拟输入，可将键鼠硬件产生的输入屏蔽掉，使之不进入 SHIQ 中，从而避免其进入应用程序的 VIQ 中，对测试造成干扰。

Windows 提供了这种机制，使用 SetWindowsHookEx 来设置低级键盘钩子（WH_KEYBOARD_LL）回调函数和低级鼠标钩子（WH_MOUSE_LL）回调函数，然后在相应的回调函数中对不同来源的事件进行过滤，从而达成目标。键盘低级钩子的回调函数代码如下：

```
LRESUL T CALLBACK myLowKeyboardProc (int code,
WPARAM wParam, LPARAM lParam )
{
KBDLLHOOKSTRUCT *ps = (KBDLLHOOKSTRUCT *)lParam;
/* 如果 code 小于零，必须立即调用 CallNextHookEx 进行处
理,并返回其结果。*/
    if(code <0)
        return CallNextHookEx (g_OldLLKeyBoardHook,
code, wParam, lParam);
```

```
/* LLKHF_INJECTED 标志，表示该事件消息是系统函数 SendInput 或 keyboard_event 模拟输入的，而不是键盘硬件的驱动程序产生的。*/
    if(ps->flags &LLKHF_INJECTED){
    OutputDebugString("man injected keyboard_event\n");
/* 如果是系统函数模拟操作产生的输入，提交到 SHIQ 中，最终提交到程序的 VIQ 中。如果是键盘硬件的驱动程序产生的输入，直接返回 1，表示不再进一步的处理，从而废弃该输入事件。*/
        return CallNextHookEx (g_OldLLKeyBoardHook, code, wParam, lParam);
    }else
        return 1;
    return CallNextHookEx (g_OldLLKeyBoardHook, code, wParam, lParam);
}
```

鼠标低级钩子回调函数类似，只是判断标志变为 LLMHF_INJECTED，可以区分是鼠标硬件产生的事件，还是 mouse_event、SendInput 产生的事件。

5.2 鼠标和键盘操作的阻断

特殊情况下可使用 BlockInput 系统函数来阻断所有的鼠标键盘输入事件。需要注意的是，BlockInput 函数不会影响异步键盘输入状态表，也就是说，当输入被阻塞时，调用 SendInput 函数仍然会改变异步键盘输入状态表。

当键鼠模拟操作发送失败了，SendInput () 函数会返回 0，表明输入操作被另一个线程阻断了，可使用 GetLastError 函数来获取错误信息。但若是 UIPI 阻止的，这种情况在 GetLastError 或返回值中是反映不出来的。

5.3 UIPI 用户界面特权隔离

UIPI 是用户界面特权隔离的简称，是 Windows 操作系统从 Vista 开始引入的一个安全特性，即特权等级低的进程不能向特权等级高的进程发送事件消息。

可通过调用 user32.dll 导出的系统函数 ChangeWindowMessageFilterEx 修改指定窗口的 UIPI 消息过滤设置，以允许发送期望的消息：

```
CHANGEFILTERSTRUCT chfit = { sizeof (CHANGEFILTERSTRUCT) };
CallChangeWindowMessageFilter (hWnd, WM_COPYDATA, MSGFLT_ALLOW, &chfit);
```

第一个参数要改变设置的窗口句构，第二个参数要过滤的消息，第三个参数要执行的动作，第四个参数指向一个 CHANGEFILTERESTRUCT 结构，用于返回修改结果。

6 结语

本研发项目在多个被测产品的测试实践中，在需要大量组合参数需要输入的功能测试中，以及功能需要反复执行高强度测试中，大大提高了缺陷的发现率和解决率。尤其是一些用户现场长期运行的生产系统出现的疑难缺陷，往往需要以月计数的时间才能在现场出现一次，查找定位问题十分棘手，耗时巨大。使用本研发项目后，可以每天 24 小时不停重复执行排查功能，往往一两天就可以重现现场疑难问题，大大加快了问题的解决。

参考文献

[1] Mark E. Russinovich and David A. Solomon with Alex Ionescu. 《Windows Internals Fifth Edition》, 2009.

[2] Microsoft. Windows Development Reference. http://msdn.microsoft.com/en-us/library/default.aspx.

（收稿日期：2012-11-12）

（上接第 64 页）

了人民币号码的快速识别。输入图像和识别结果如图 9 所示。

图 9 识别的结果

对多个样本图像进行测试，均能快速识别，识别准确率较高。

4 结语

对人民币纸币号码进行分析，提出了依据大写字母和数字的拓扑结构特征和端点特征的识别方法，该方法较简单可行，具有效率高、速度快、抗噪能力强等特点，但也有误识别情形，如“5”误识为“S”，“O”误识为“0”等，由于纸币质量不高，导致预处理提取的信息不准确，引起误识，有待今后进一步改进。该方法在车牌识别的应用中有着广泛的前景。对汉字手写体识别的研究有一定的借鉴和指导作用。

参考文献

[1] 刘家锋，刘松波，唐降龙.一种实时纸币识别方法的研究 [J]．计算机研究与发展，2003，40（7）：1057-1061.

[2] 王惠，鲁五一．基于 ARM 的纸币号码识别系统 [J]．计算技术与自动化，2010，29（2）：81-85.

[3] 陆宗骐，金登男．Visual C++.NET 图像处理编程 [M]．北京：清华大学出版社，2006.

[4] 吕凤军．数字图象处理编程入门 [M]．北京：清华大学出版社，1999.

（收稿日期：2012-10-26）

C# 语言开发俄罗斯方块游戏

王文举

摘　要：介绍利用 C# 语言开发俄罗斯方块游戏的设计方法和代码实现，对开发类似的 GDI+游戏有很好的借鉴作用。

关键词：C# 语言；俄罗斯方块游戏；GDI+游戏

1　引言

俄罗斯方块游戏是一种大家较为熟悉的游戏，通常有 7 种方块模型，可以控制它的平移、加速、旋转变形（7 种模型共计 19 种状态）。另外，左右平移和变形需要考虑边界碰撞的问题，下移需要考虑堆积和消层的问题。

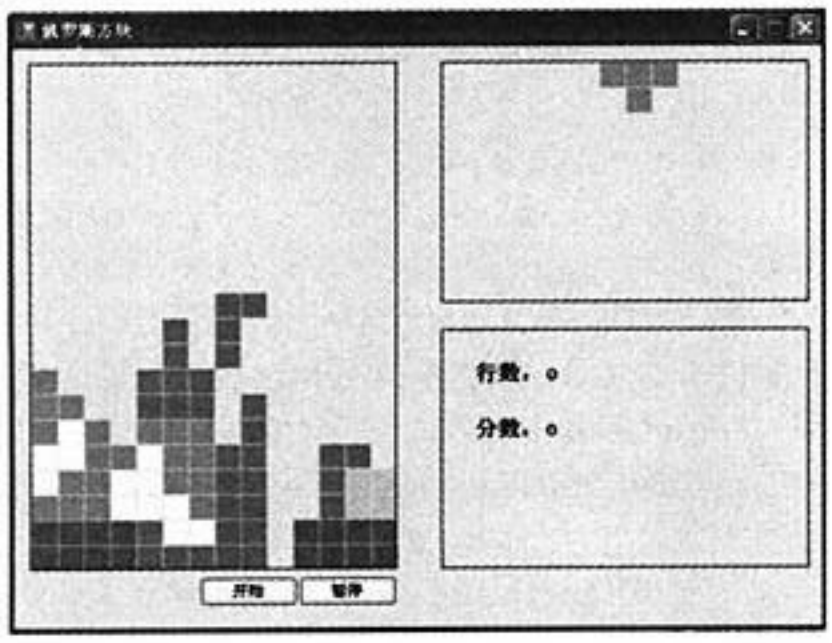


图 1　俄罗斯方块游戏界面示例

2　设计思路

利用 C# 语言开发了俄罗斯方块游戏，如图 1 所示，设计中借鉴了 MVC 的设计思路，除了窗口界面类外设计了 3 个类：

一是模型类（CakeMode），功能为实现模型的构造（CakeMode (int _mode)），获取方块移动后的位置（Point [] CakeMove (int n)），获取方块旋转变形后的位置（Point [] CakeChange ()）。

二是游戏规则类（Rule），功能为判断方块移动后的位置是否超出边界（bool MoveStop (Point [] ArryMveNext,int n)），判断方块旋转变形后的位置是否超出边界（bool ChangeStop (Point [] ArryChgNext)），去除已添满的行并加分（void RefurbishRow (int Max, int Min))。

三是画图类（Draw），功能为绘制方块的各个子方块（void DrawCake (Point [] ArryPoi,Color color)），清空当前方块的区域（void CakeDelete (Point [] ArryPoi)），清空游戏背景（void BackgroundClear ()），重绘游戏背景上的方块（void DrawBackground ()）。

该方法通过模型、控制、视图相分离，使程序流程更加清晰，并且具有强内聚，松耦合特性，使功能扩展更加方便。该方法对于开发 GDI+游戏有很好的借鉴作用。

3　实现代码

3.1　窗口界面类

```
    public partial class Form1 : Form
    {   CakeMode myCake;CakeMode myCakeNext;
//pictureBox1 和 pictureBox3 中方块模型
            Draw myDraw; Draw myDrawNext; //pictureBox1
//和 pictureBox3 中画图
        rule myRule;//动作规则
        public static int conMax = 0;//方块落下后的最大位置
        public static int conMin = 0;//方块落下后的最小位置
    public static int CakeNO = 0;//记录下一个方块模型的标识
        public static bool become = false;//判断是否生成下一个
//方块模型
    public static bool isbegin = false;//判断当前游戏是否开始
        public bool ispause = true;//判断是否暂停游戏
        Point[] ArryMveNext; //方块移动后下一个位置
        Point[] ArryChgNext; //方块旋转变形后下一个位置
        public Form1()
        { InitializeComponent(); timer1.Enabled = false;}
        //按键"开始"的响应函数
        private void button1_Click(object sender, EventArgs e)
        {   myDraw = new Draw(pictureBox1);
            myRule = new rule(pictureBox1);
            myDrawNext = new Draw(pictureBox3);
            myDraw.BackgroundClear();//清空整个控件
            Random rand = new Random();//实例化 Random
            CakeNO = rand.Next(1, 8);//获取随机数
      myCake = new CakeMode(CakeNO);//设置方块的模型
            label3.Text = "0";//显示去除的行数
            label4.Text = "0";//显示分数
            myRule.Label_Linage = label3;//将 label3 控件加载到
//Russia 类中
```

```
        myRule.Label_Fraction = label4;//将 label4 控件加载
//到 Russia 类中
        myDraw.DrawCake(myCake.ArryPoi, myCake.color);
//绘制组合方块
        NextCake();//pictureBox3 中生成下一个方块模型
        isbegin = true; ispause = true;
        button2.Text = "暂停";
        timer1.Enabled = true;  timer1.Start();
        textBox1.Focus();//获取焦点,是为了绑定键盘事件
    }
    //计时器的响应事件
    private void timer1_Tick(object sender, EventArgs e)
    {  //方块下移
    CakeMove(0);
    if (become)//如果显示新的方块
    {  NextCake(); become = false; }
    textBox1.Focus();//获取焦点,是为了绑定键盘事件
    }
    //pictureBox3 中生成下一个方块模型
    public void NextCake()
    {  myDrawNext.BackgroundClear();
    Random rand = new Random();//实例化 Random
    CakeNO = rand.Next(1, 8);//获取随机数
    myCakeNext = new CakeMode(CakeNO);
        myDrawNext.DrawCake    (myCakeNext.ArryPoi,
myCakeNext.color);//绘制组合方块
      }
      //键盘的响应事件,利用 Up,Down,Left,Right 四个键盘按
//键对方块进行控制
          private void Form1_KeyDown (object sender,
KeyEventArgs e)
      {  if (! isbegin) return;
         if (! ispause) return;
         if (e.KeyCode == Keys.Up)//如果当前按下的是↑键
            CakeChange();//方块变形
    if (e.KeyCode == Keys.Down)//如果当前按下的是↓键
    { timer1.Interval = 100; CakeMove(0);} //方块加速下移
         if (e.KeyCode == Keys.Left)//如果当前按下的是←键
            CakeMove(1);//方块左移
    if (e.KeyCode == Keys.Right)//如果当前按下的是→键
            CakeMove(2);//方块右移
      }
          private void Form1_KeyUp (object sender,
KeyEventArgs e)
      {  if (! isbegin) return;
         if (! ispause) return;
    if (e.KeyCode == Keys.Down)//如果当前松开的是↓键
            {timer1.Interval = 500;//恢复下移的速度 }
         textBox1.Focus();//获取焦点,是为了绑定键盘事件
      }
      //方块变形的事件
      private void CakeChange()
      {  ArryChgNext = myCake.CakeChange();//获取方块变
//形后的位置
          bool tem_bool = myRule.ChangeStop(ArryChgNext);
//判断方块变形后是否出边界
          if (tem_bool)//如果变形后没有出边界
          {  myDraw.CakeDelete(myCake.ArryPoi);//清空当前
//方块的区域
              //获取变形后方块的位置
    for (int i = 0; i < myCake.ArryChgNext.Length; i++)
                myCake.ArryPoi[i] = myCake.ArryChgNext[i];
              myCake.firstPoi = myCake.ArryPoi[0];//记录方块
//的起始位置
                myDraw.DrawCake    (myCake.ArryChgNext,
myCake.color);//绘制变形后方块
          }
      }
      //方块移动的事件
      private void CakeMove(int n)
      {  ArryMveNext = myCake.CakeMove(n); //获取方块移
//动后的位置
          bool tem_bool = myRule.MoveStop (myCake.
ArryMveNext, n);//判断方块移动后是否出边界
          if (tem_bool)//如果移动后没有出边界
          {  myDraw.CakeDelete(myCake.ArryPoi);//清空当前
//方块的区域
              //获取移动后方块的位置
        for (int i = 0; i < myCake.ArryMveNext.Length; i++)
              myCake.ArryPoi[i] = myCake.ArryMveNext[i];
            myCake.firstPoi = myCake.ArryPoi[0];//记录方块
//的起始位置
                myDraw.DrawCake    (myCake.ArryMveNext,
myCake.color);//绘制移动后方块
          }
          else//如果方块到达底部
          {  if (! tem_bool && n == 0)//如果当前方块是下移
             {  conMax = 0;//记录方块落下后的顶端位置
              conMin = pictureBox1.Height;//记录方块落下
//后的底端位置
              //遍历方块的各个子方块
              for (int i = 0; i < myCake.ArryPoi.Length; i++)
              {  if (myCake.ArryPoi[i].Y < rule.maxY)//记录方
//块的顶端位置
                  rule.maxY = myCake.ArryPoi[i].Y;
                //记录指定的位置已存在方块
                rule.Place  [myCake.ArryPoi  [i].X  /  20,
myCake.ArryPoi[i].Y / 20] = true;
                //记录方块的颜色
          rule.PlaceColor  [myCake.ArryPoi  [i].X / 20, myCake.
ArryPoi[i].Y / 20] = myCake.color;
      if (myCake.ArryPoi[i].Y > conMax)//记录方块的顶端位置
                  conMax = myCake.ArryPoi[i].Y;
              if (myCake.ArryPoi[i].Y < conMin)//记录方块
```

GAME PROGRAM

```
//的底端位置
                    conMin = myCake.ArryPoi[i].Y;
                }
                if (myCake.firstPoi.X == 140 && myCake.
firstPoi.Y == 20)
                { timer1.Stop();isbegin = false; return; }
                myCake = myCakeNext; // pictureBox3 中的下
//一个模型开始成为当前模型
                myDraw.DrawCake(myCake.ArryPoi, myCake.
color);//绘制组合方块
                myRule.RefurbishRow(conMax, conMin);//去除
//已填满的行
        become = true;//标识,判断可以生成下一个方块
            }
        }
    }
    //按键"暂停"响应函数
    private void button2_Click(object sender, EventArgs e)
    {   if (button2.Text == "暂停")
        {timer1.Stop(); button2.Text = "继续";ispause = false;
textBox1.Focus();}
        else
        {timer1.Start();button2.Text = "暂停";ispause = true;
textBox1.Focus();}
    }
        //pictureBox1 中游戏背景刷新事件
        private void pictureBox1_Paint (object sender,
PaintEventArgs e)
    {   if (isbegin) {  myDraw.DrawBackground(); } }
  }
```

3.2 CakeMode 类

```
  class CakeMode
  {  public Point firstPoi = new Point(140, 20);//定义方块的
//起始位置
    public Point[] ArryPoi = new Point[4];//方块的数组
    public Point[] ArryMveNext = new Point[4];//移动后下
//一个方块的数组
    public Point[] ArryChgNext = new Point[4];//变形后下一
//个方块的数组
        int Cake = 20;//定义方块的大小
 public int mode;//模型型号    public Color color; //模型颜色
    public int state_n;//记录方块的下一个变形样式
    public CakeMode(int _mode)
    {  mode = _mode ;
       ArryPoi[0] = firstPoi;//记录方块的起始位置
       switch (mode)//根据参数构造模型,生成俄罗斯方块
       {  case 1://组合"L"方块
            {
          ArryPoi [1] = new Point (firstPoi.X, firstPoi.Y -
Cake);//设置第二块方块的位置
              ArryPoi[2] = new Point(firstPoi.X, firstPoi.Y +
Cake);//设置第三块方块的位置
          ArryPoi [3] = new Point (firstPoi.X + Cake, firstPoi.Y +
Cake);//设置第四块方块的位置
                color = Color.Fuchsia;//设置当前方块的颜色
          state_n = 2;//记录方块的下一个变形样式  break; }
            case 2://组合"Z"方块
              {  ArryPoi[1] = new Point(firstPoi.X, firstPoi.Y
- Cake);
                 ArryPoi [2] = new Point (firstPoi.X - Cake,
firstPoi.Y - Cake);
                 ArryPoi [3] = new Point (firstPoi.X + Cake,
firstPoi.Y);
          color = Color.Yellow; state_n = 6; break;      }
            case 3://组合倒"L"方块
       {  ArryPoi[1] = new Point(firstPoi.X, firstPoi.Y - Cake);
          ArryPoi[2] = new Point(firstPoi.X, firstPoi.Y + Cake);
                 ArryPoi [3] = new Point (firstPoi.X - Cake,
firstPoi.Y + Cake);
        color = Color.CornflowerBlue;state_n = 8;break; }
            case 4://组合倒"Z"方块
    {  ArryPoi[1] = new Point(firstPoi.X, firstPoi.Y - Cake);
                 ArryPoi [2] = new Point (firstPoi.X + Cake,
firstPoi.Y - Cake);
                 ArryPoi [3] = new Point (firstPoi.X - Cake,
firstPoi.Y);
          color = Color.Blue; state_n = 12;break;        }
            case 5://组合"T"方块
              {  ArryPoi[1] = new Point(firstPoi.X, firstPoi.Y
- Cake);
                 ArryPoi [2] = new Point (firstPoi.X + Cake,
firstPoi.Y - Cake);
                 ArryPoi [3] = new Point (firstPoi.X - Cake,
firstPoi.Y - Cake);
          color = Color.DarkSalmon ;state_n = 14;break;    }
            case 6://组合"一"方块
       {  ArryPoi[1] = new Point(firstPoi.X + Cake, firstPoi.Y);
        ArryPoi[2] = new Point(firstPoi.X - Cake, firstPoi.Y);
      ArryPoi[3] = new Point(firstPoi.X - Cake * 2, firstPoi.Y);
          color = Color.Red; state_n = 18; break;        }
            case 7://组合"田"方块
       {  ArryPoi[1] = new Point(firstPoi.X - Cake, firstPoi.Y);
                 ArryPoi [2] = new Point (firstPoi.X - Cake,
firstPoi.Y - Cake);
        ArryPoi[3] = new Point(firstPoi.X, firstPoi.Y - Cake);
          color = Color.LightGreen; state_n = 19; break;    }
        }
    }
    //方块移动,返回移动后的位置
    public Point[] CakeMove(int n)
    {  //将当前方块的位置存入到移动后方块数组中
       for (int i = 0; i < ArryMveNext.Length; i++)
          ArryMveNext[i] = ArryPoi[i];
```

```
        switch (n)//方块的移动方向
        { case 0://下移
       { //遍历方块中的子方块,使各子方块下移一个方块位
                  for (int i = 0; i < ArryMveNext.Length; i++)
                 ArryMveNext[i] = new Point(ArryMveNext[i].
X, ArryMveNext[i].Y + Cake);
                  break; }
             case 1://左移
                { for (int i = 0; i < ArryMveNext.Length; i++)
                  ArryMveNext [i] = new Point(ArryMveNext
[i].X – Cake, ArryMveNext[i].Y);
                  break; }
             case 2://右移
                { for (int i = 0; i < ArryMveNext.Length; i++)
                  ArryMveNext[i] = new Point(ArryMveNext
[i].X + Cake, ArryMveNext[i].Y);
                  break; }
        }
        return ArryMveNext;
      }
      //方块变形,返回旋转变形后的位置
      public Point[] CakeChange()
      { //将当前方块的位置存入到变形后方块数组中
        for (int i = 0; i < ArryChgNext.Length; i++)
          ArryChgNext[i] = ArryPoi[i];
        switch (mode)//根据模型的型号
        { case 1://"L"方块
            if (state_n == 1) //选择该模型的变形样式
             { ArryChgNext [1] = new Point (firstPoi.X,
firstPoi.Y – Cake);
               ArryChgNext [2] = new Point (firstPoi.X,
firstPoi.Y + Cake);
               ArryChgNext [3] = new Point (firstPoi.X +
Cake, firstPoi.Y + Cake);
               state_n = 2; break; }
            if (state_n == 2)
             { ArryChgNext [1] = new Point (firstPoi.X –
Cake, firstPoi.Y);
               ArryChgNext [2] = new Point (firstPoi.X +
Cake, firstPoi.Y);
               ArryChgNext [3] = new Point (firstPoi.X +
Cake, firstPoi.Y – Cake);
               state_n = 3; break;}
            if (state_n == 3)
             { ArryChgNext [1] = new Point (firstPoi.X,
firstPoi.Y – Cake);
               ArryChgNext [2] = new Point (firstPoi.X –
Cake, firstPoi.Y – Cake);
               ArryChgNext [3] = new Point (firstPoi.X,
firstPoi.Y + Cake);
               state_n = 4; break; }
            if (state_n == 4)
             { ArryChgNext [1] = new Point (firstPoi.X +
Cake, firstPoi.Y);
               ArryChgNext [2] = new Point (firstPoi.X –
Cake, firstPoi.Y);
               ArryChgNext [3] = new Point (firstPoi.X –
Cake, firstPoi.Y + Cake);
               state_n = 1; break; }
            break;
          case 2://Z
            if (state_n == 5)
             { ArryChgNext [1] = new Point (firstPoi.X,
firstPoi.Y – Cake);
               ArryChgNext [2] = new Point (firstPoi.X –
Cake, firstPoi.Y – Cake);
               ArryChgNext [3] = new Point (firstPoi.X +
Cake, firstPoi.Y);
               state_n = 6; break;}
            if (state_n == 6)
             { ArryChgNext [1] = new Point (firstPoi.X +
Cake, firstPoi.Y);
               ArryChgNext [2] = new Point (firstPoi.X +
Cake, firstPoi.Y – Cake);
               ArryChgNext [3] = new Point (firstPoi.X,
firstPoi.Y + Cake);
               state_n = 5; break;}
            break;
          case 3://倒 L
            if (state_n == 7)
             { ArryChgNext [1] = new Point (firstPoi.X,
firstPoi.Y – Cake);
               ArryChgNext [2] = new Point (firstPoi.X,
firstPoi.Y + Cake);
               ArryChgNext [3] = new Point (firstPoi.X –
Cake, firstPoi.Y + Cake);
               state_n = 8; break;}
            if (state_n == 8)
             { ArryChgNext [1] = new Point (firstPoi.X –
Cake, firstPoi.Y);
               ArryChgNext [2] = new Point (firstPoi.X +
Cake, firstPoi.Y);
               ArryChgNext [3] = new Point (firstPoi.X +
Cake, firstPoi.Y + Cake);
               state_n = 9; break; }
            if (state_n == 9)
             { ArryChgNext [1] = new Point (firstPoi.X,
firstPoi.Y – Cake);
               ArryChgNext [2] = new Point (firstPoi.X,
firstPoi.Y + Cake);
               ArryChgNext [3] = new Point (firstPoi.X +
Cake, firstPoi.Y – Cake);
               state_n = 10; break; }
            if (state_n == 10)
```

GAME PROGRAM

```
                { ArryChgNext [1] = new Point (firstPoi.X –
Cake, firstPoi.Y);
                  ArryChgNext [2] = new Point (firstPoi.X +
Cake, firstPoi.Y);
                  ArryChgNext [3] = new Point (firstPoi.X –
Cake, firstPoi.Y – Cake);
                  state_n = 7; break; }
               break;
            case 4://倒 Z
               if (state_n == 11)
                { ArryChgNext [1] = new Point (firstPoi.X,
firstPoi.Y – Cake);
                  ArryChgNext [2] = new Point (firstPoi.X +
Cake, firstPoi.Y – Cake);
                  ArryChgNext [3] = new Point (firstPoi.X –
Cake, firstPoi.Y);
                  state_n = 12; break; }
               if (state_n == 12)
                { ArryChgNext [1] = new Point (firstPoi.X –
Cake, firstPoi.Y);
                  ArryChgNext [2] = new Point (firstPoi.X –
Cake, firstPoi.Y – Cake);
                  ArryChgNext [3] = new Point (firstPoi.X,
firstPoi.Y + Cake);
                  state_n = 11; break; }
               break;
            case 5://T
               if (state_n == 13)
                { ArryChgNext [1] = new Point (firstPoi.X,
firstPoi.Y – Cake);
                  ArryChgNext [2] = new Point (firstPoi.X +
Cake, firstPoi.Y – Cake);
                  ArryChgNext [3] = new Point (firstPoi.X –
Cake, firstPoi.Y – Cake);
                  state_n = 14; break; }
               if (state_n == 14)
                { ArryChgNext [1] = new Point (firstPoi.X,
firstPoi.Y – Cake);
                  ArryChgNext [2] = new Point (firstPoi.X,
firstPoi.Y + Cake);
                  ArryChgNext [3] = new Point (firstPoi.X +
Cake, firstPoi.Y);
                  state_n = 15; break; }
               if (state_n == 15)
                { ArryChgNext [1] = new Point (firstPoi.X –
Cake, firstPoi.Y);
                  ArryChgNext [2] = new Point (firstPoi.X +
Cake, firstPoi.Y);
                  ArryChgNext [3] = new Point (firstPoi.X,
firstPoi.Y – Cake);
                  state_n = 16; break; }
               if (state_n == 16)
                { ArryChgNext [1] = new Point (firstPoi.X,
firstPoi.Y – Cake);
                  ArryChgNext [2] = new Point (firstPoi.X –
Cake, firstPoi.Y);
                  ArryChgNext [3] = new Point (firstPoi.X,
firstPoi.Y + Cake);
                  state_n = 13; break; }
               break;
            case 6://一
               if (state_n == 17)
                { ArryChgNext [1] = new Point (firstPoi.X +
Cake, firstPoi.Y);
                  ArryChgNext [2] = new Point (firstPoi.X –
Cake, firstPoi.Y);
                  ArryChgNext [3] = new Point (firstPoi.X –
Cake * 2, firstPoi.Y);
                  state_n = 18; break;}
               if (state_n == 18)
                { ArryChgNext [1] = new Point (firstPoi.X,
firstPoi.Y – Cake);
                  ArryChgNext [2] = new Point (firstPoi.X,
firstPoi.Y + Cake);
                  ArryChgNext [3] = new Point (firstPoi.X,
firstPoi.Y + Cake * 2);
                  state_n = 17; break;}
               break;
            case 7://田
               if (state_n == 19)
                { ArryChgNext [1] = new Point (firstPoi.X –
Cake, firstPoi.Y);
                  ArryChgNext [2] = new Point (firstPoi.X –
Cake, firstPoi.Y – Cake);
                  ArryChgNext [3] = new Point (firstPoi.X,
firstPoi.Y – Cake);
                  state_n = 19; break; }
               break;
         }
         return ArryChgNext;
      }
   }
```

3.3 Rule 类

```
   class rule
   { public static int conWidth = 0;//记录列数
     public static int conHeight = 0;//记录行数
     public static Color[,] PlaceColor;//记录方块的颜色
     public static bool[,] Place;//记录方块的位置
     public static int maxY = 0;//方块在行中的最大高度
     public static int conMax = 0;//方块落下后的最大位置
     public static int conMin = 0;//方块落下后的最小位置
       bool[] tem_Array = { false, false, false, false };//记录方
//块组中那一块所在行中已满
```

```
        public Label Label_Linage = new Label();//实例化
//Label,用于显示去除的行数
        public Label Label_Fraction = new Label();//实例化
//Label,用于显示分数
    public static int[] ArrayCent = new int[] { 2, 5, 9, 15 };
//记录加分情况
    int Cake = 20;//定义方块的大小
    Control Mycontrol;
    public rule(Control _mycontrol)
    {  Mycontrol = _mycontrol;
  conWidth = Mycontrol.Width / 20;//获取背景的总行数
  conHeight = Mycontrol.Height / 20;//获取背景的总列数
        Place = new bool[conWidth, conHeight];//定义记录
//各方块位置的数组
        PlaceColor = new Color[conWidth, conHeight];//定义
//记录各方块颜色的数组
        //对各方块的信息进行初始化
        for (int i = 0; i < conWidth; i++)
        {  for (int j = 0; j < conHeight; j++)
           {  Place[i, j] = false;//方块为空
    PlaceColor[i, j] = Mycontrol.BackColor;//与背景色相同
           }
        }
        maxY = conHeight * Cake;//记录方块的最大值
    }
    //判断方块移动时是否出边界,返回 true 为不超出边界
    public bool MoveStop(Point[] ArryMveNext,int n)
    {  bool tem_bool = true;
       int tem_width = 0;
       int tem_height = 0;
       switch (n)
       { case 0://下移
             {  //遍历方块中的各个子方块
                for (int i = 0; i < ArryMveNext.Length; i++)
                {  tem_width = ArryMveNext[i].X / 20;
//获取方块的横向坐标值
                   tem_height = ArryMveNext[i].Y / 20;
//获取方块的纵向坐标值
          //判断是否超出底边界,或是与其他方块重叠
                   if  (tem_height == conHeight || Place
[tem_width, tem_height])
                      tem_bool = false;//超出边界
                } break;
             }
          case 1://左移
             {  for (int i = 0; i < ArryMveNext.Length; i++)
                {  tem_width = ArryMveNext[i].X / 20;
                   tem_height = ArryMveNext[i].Y / 20;
    if (tem_width == -1 || Place[tem_width, tem_height])
                      tem_bool = false;
                } break;
             }
          case 2://右移
             {  for (int i = 0; i < ArryMveNext.Length; i++)
                {  tem_width = ArryMveNext[i].X / 20;
                   tem_height = ArryMveNext[i].Y / 20;
                    if  (tem_width == conWidth || Place
[tem_width, tem_height])
                      tem_bool = false;
                } break;
             }
             return tem_bool;
       }
    //判断方块变形时是否出边界,返回 true 为不超出边界
       public bool ChangeStop(Point[] ArryChgNext)
       {  bool tem_bool = true;//判断方块是否可变
          //遍历方块的各个子方块
          for (int i = 0; i < ArryChgNext.Length; i++)
    {  if (ArryChgNext[i].X / 20 < 0)//变换后是否超出左边界
             {  tem_bool = false; break;}
             if (ArryChgNext[i].X / 20 >= conWidth)//变换后是
//否超出右边界
             {  tem_bool = false; break;}
             if (ArryChgNext[i].Y / 20 >= conHeight)//变换后是
//否超出下边界
             {  tem_bool = false; break;}
             //变换后是否与其他方块重叠
       if (Place[ArryChgNext[i].X / 20, ArryChgNext[i].Y / 20])
             {  tem_bool = false; break; }
          }
          return tem_bool;
       }
       //去除已添满的行
       public void RefurbishRow(int Max, int Min)
       {  int tem_max = Max / 20;//获取方块的最大位置在多
//少行
    int tem_min = Min / 20;//获取方块的最小位置在多少行
          bool tem_bool = false;
          //初始化记录刷新行的数组
          for (int i = 0; i < tem_Array.Length; i++)
             tem_Array[i] = false;
          int state_n = maxY;//记录最高行的位置
    for (int i = 0; i < 4; i++)//查找要刷新的行
    {  if ((tem_min + i) > 19)//如果超出边界
             break;//退出本次操作
       tem_bool = false;
       //如果当前行中有空格
       for (int k = 0; k < conWidth; k++)
       {  if (! Place[k, tem_min + i])//如果当前位置为空
          {  tem_bool = true; break;  }
       }
       if (! tem_bool)//如要当行为满行
       {  tem_Array[i] = true;//记录为刷新行 }
```

```
        }
        int Progression = 0;//记录去除的几行
        if  (tem_Array [0] == true || tem_Array [1] == true ||
tem_Array[2] == true || tem_Array[3] == true)//如果有刷新行
        {  int Trow = 0;//记录最小行数
           for (int i = (tem_Array.Length - 1); i >= 0; i--)
//遍历记录刷新行的数组
           {  if (tem_Array[i])//如果是刷新行
              {  Trow = Min / 20 + i;//记录最小行数
                 //将刷新行到背景顶端的区域下移
                 for (int j = Trow; j >= 1; j--)
                 {  for (int k = 0; k < conWidth; k++)
                    {  PlaceColor[k, j] = PlaceColor[k, j - 1];
//记录方块的颜色
                       Place[k, j] = Place[k, j - 1];//记录方块的位置
                    }
                 }
              Min += 20;//方块的最小位置下移一个方块位
                 //将背景的顶端清空
                 for (int k = 0; k < conWidth; k++)
                 {  PlaceColor[k, 0] = Mycontrol.BackColor;
//记录方块的颜色
                    Place[k, 0] = false;//记录方块的位置
                 }
                 Progression += 1;//记录刷新的行数
              }
           }
           Graphics g = Mycontrol.CreateGraphics();
           for (int i = 0; i < conWidth; i++)
           {  for (int j = 0; j <= Max / 20; j++)
              {  //获取各方块的区域
                 Rectangle rect = new Rectangle(i * Cake +
1, j * Cake + 1, 19, 19);
                 //绘制已落下的方块
              g.FillRectangle(new SolidBrush(PlaceColor[i, j]), rect);
              }
           }
           //显示当前的刷新行数
        Label_Linage.Text = Convert.ToString (Convert.ToInt32
(Label_Linage.Text) + Progression);
           //显示当前的得分情况
           Label_Fraction.Text = Convert.ToString (Convert.
ToInt32(Label_Fraction.Text) + ArrayCent[Progression - 1]);
        }
     }
  }
```

3.4 Draw 类

```
  class Draw
  {  Control Mycontrol;  Graphics g;
     public Draw(Control control)
     { Mycontrol = control; g = control.CreateGraphics();}
     //绘制方块的各个子方块
     public void DrawCake(Point[] ArryPoi,Color color)
     {  for (int i = 0; i < ArryPoi.Length; i++)
        {  //获取子方块的区域,用 color 填充
     Rectangle rect = new Rectangle(ArryPoi[i].X + 1, ArryPoi
[i].Y + 1, 19, 19); g.FillRectangle(new SolidBrush(color), rect);
        }
     }
     //清空当前方块的区域
     public void CakeDelete(Point[] ArryPoi)
     {  for (int i = 0; i < ArryPoi.Length; i++)//遍历方块的各
//个子方块
        {  //获取各子方块的区域,用背景色填充背景
           Rectangle rect = new Rectangle (ArryPoi [i].X,
ArryPoi[i].Y, 20, 20);
           g.FillRectangle (new SolidBrush (Mycontrol .
BackColor), rect);
        }
     }
     //清空游戏背景
     public void BackgroundClear()
     {  if (Mycontrol ! = null)//如要已载入背景控件
        { Rectangle rect = new Rectangle(0, 0, Mycontrol.
Width, Mycontrol.Height);
           g.FillRectangle (new SolidBrush (Mycontrol.
BackColor), rect);
        }
     }
     //重绘游戏背景上的方块
     public void DrawBackground()
     {  for (int i = 0; i <= (Mycontrol.Width / 20 - 1); i++)
   {  for (int j = 0; j <= (Mycontrol.Height / 20 - 1); j++)
        {  //获取各方块的绘制区域
           Rectangle rect = new Rectangle(i * 20 + 1, j *
20 + 1, 19, 19);
           g.FillRectangle(new SolidBrush(rule.PlaceColor
[i, j]), rect);
        }
     }
   }
```

4 结语

利用 C# 开发俄罗斯方块游戏，借鉴了 MVC 的设计模式，通过建立模型类、游戏规则类和画图类，使程序流程更加清晰，并且程序具有强内聚，松耦合特性，使功能扩展更加方便。该方法对于开发类似的 GDI+游戏有很好的借鉴价值。

（收稿日期：2012-09-27）

基于PB的多版本PHP集成开发环境系统设计与开发

张瑞祥

摘　要：通过相应服务组件提供命令与PowerBuilder程序编写，高度集成PHP多个主流版本、Apache服务组件、MySQL服务组件和第三方常用工具，主要解决一键配置环境、多版本PHP程序切换和可以利用移动存储设备移动运行的问题，并且让测试人员和开发人员任意选择PHP程序版本进行开发工作，从而提高了工作效率。

关键词：PHP语言；集成环境；多版本

1　引言

随着信息技术的高速发展，PHP语言在不同领域开发的B/S架构程序中得到广泛使用。越来越多的程序员使用PHP语言进行程序开发。在使用自己搭建的PHP环境或集成的PHP开发环境，总是有一个问题长期困扰着程序员，那就是PHP的版本之前存在差异，可能会导致程序在不同的PHP版本中产生不必要的BUG。为了程序提高对PHP运行环境兼容性，很多人都会在不同PC机器搭建环境或者在同一台机器配置多个环境手动修改配置文件来解决这个问题。所以设计一款能够一键配置并且集成多个版本的PHP开发环境是非常必要的，这也是本系统设计的初衷。

2　需求分析

根据PHP官方资料和国内云计算开放平台资料的分析，现在PHP主要使用开发的版本PHP5.2.17、PHP5.3.5和PHP5.4.0。通过网络使用比较多了几款WAMP集成开发环境分析，PHP集成开发环境需要完成对PHP、Apache、MYSQL和相关实用工具的集成。

本系统设计集成了PHP5.2.17、PHP5.3.5、PHP5.4.0、MYSQL5.1、Apache2.2.21及相关实用工具，并且实现一键切换PHP版本、自动生成基本配置文件、服务状态管理和通过移动存储设备做到移动开发的功能。

3　开发工具及必要运行服务组件选择与下载

3.1　开发工具介绍

本系统采用PB（PowerBuilder的缩写）作为开发工具。PB是一款可视化、多特性的开发工具，它的语言是PowerScript。PowerScript是最新第四代语言——面向对象语言。

3.2　运行服务组件选择与下载

3.2.1 Web服务器组件

本系统使用Apache作为服务器组件，可以通过httpd.apache.org下载Window版本的Apache发行包，也可以下载源码版自行使用VC6.0进行编译，但是编译难度较大，建议还是选择发行包。Apache是Apache HTTP Server的简称，它是世界使用排名第一的Web服务器软件，是Apache软件基金会的一个开放源码的网页服务器。它几乎可以在所有广泛使用的计算机平台上运行。由于其跨平台和安全性被广泛使用，是最流行的Web服务器端软件之一。

3.2.2 PHP语言组件

本系统使用PHP5.2.17、PHP5.3.5、PHP5.4.0 3个版本的PHP语言组件，可以通过www.php.net选择PHP版本进行下载。值得注意的是PHP5.3以后有两个版本，分别是VC6和VC9，他们之间的区别VC6使用VisualStudio6进行编译，而VC9使用VisualStudio2008进行编译。这里使用Apache的Web服务器建议使用VC6的版本。PHP是英文超文本预处理语言Hypertext Preprocessor的缩写。PHP是一种HTML内嵌式的语言，是一种在服务器端执行的嵌入HTML文档的脚本语言，语言的风格有类似于C语言，被广泛地运用。它独特的语法混合了C、Java、Perl以及PHP自创的语法。 PHP安装它可以比CGI或者Perl更快速地执行动态网页。用PHP做出的动态页面与其他的编程语言相比，PHP是将程序嵌入到HTML文档中去执行，执行效率比完全生成HTML标记的CGI要高许多；PHP还可以执行编译后代码，编译可以达到加密和优化代码运行，使代码运行更快。PHP具有非常强大的功能，所有的CGI的功能PHP都能实现，而且支持几乎所有流行的数据库以及操作系统。最重要的是PHP可以用C、C++进行程序的扩展。

3.2.3 数据库组件

本系统使用MySQL5.1作为数据库组件，可以通过www.mysql.com下载，注意必须下载MySQL免安装版本的Windows程序。MySQL是一个中、小型关系型数据库管理系统，由瑞典MySQL AB公司开发，目前属于Oracle公司。MySQL的SQL语言是用于访问数据库的最常用标准化语言。MySQL软件采用

了 GPL（GNU 通用公共许可证），它分为免费版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。由于其免费版的性能卓越，搭配 PHP 和 Apache 可组成良好的开发环境。

4 系统设计

4.1 PHP 集成环境设计原理

根据 Apache 组件与 MySQL 组件提供命令接口和实际需求，分析出系统主要原理如图 1 所示。

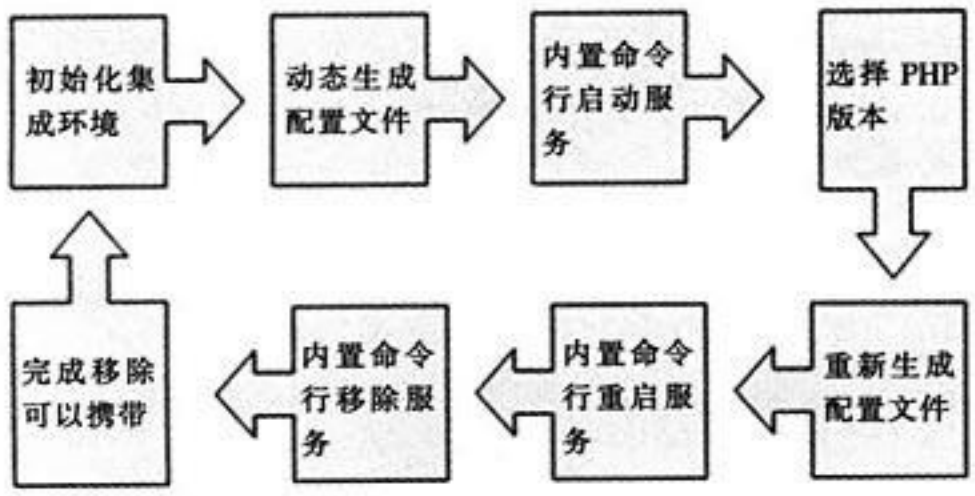


图 1 PHP 集成环境设计原理

系统原理主要通过管理程序控制集成环境的初始化，将组件组成到服务启动项中，再进行默认配置文件动态生成和使用组件内置命令启动相应服务，完成默认的集成环境配置与启动。切换 PHP 版本后，通过重新生成配置文件和使用组件内置命令重启服务完成对配置文件重新载入，完成版本切换。并通过组件内置命令移除服务，实现可移动携带的功能。

4.2 关键技术与界面设计

4.2.1 Apache 组件提供程序命令接口的说明

Apache 组件中提供 httpd 带参数命令实现 Apache 的服务注册、启动、删除、重启等操作，并且所有命令必须在 Apache 组件的 bin 目录运行才能有效。编写集成开发环境所用到的命令参数与解释如下：

（1）命令格式：httpd -k install -n “apache”。

含义解释：-k install 表示注册服务，-n “apache” 表示自定义服务名为 Apache，这里需要注意的是” apache” 是自定义的，可以自行修改，如果机器已经默认安装了 apache 服务，在使用自行开发的系统时，将会造成不必要的冲突，建议自行设定一个服务名称，且将在后续操作中反复使用。

（2）命令格式：httpd -k start -n “apache”。

含义解释：-k start 表示启动服务，-n “apache” 表示自定义服务名为 Apache。如果之前自行修改过服务名称，这里就是填写自己设定的名称，以后的操作也是相同。

（3）命令格式：httpd -k restart -n “apache”。

含义解释：-k restart 表示重启服务，-n “apache” 表示自定义服务名为 apache。

（4）命令格式：httpd -k stop -n “apache”。

含义解释：-k stop 表示停止服务，-n “apache” 表示自定义服务名为 apache。

（5）命令格式：httpd -k uninstall -n “apache”。

含义解释：-k uninstall 表示删除服务，-n “apache” 表示自定义服务名为 apache。在执行删除服务操作前，必须先将 Apache 服务停止后才能移除。

通过以上 5 条常用命令就可以完成 Apache 的服务注册、启动、删除、重启基本操作。

4.2.2 Mysql 组件提供程序命令接口的说明

Apache 组件中提供 mysqld 和 net 带参数命令实现 Mysql 的服务注册、启动、删除、重启等操作，并且所有命令必须在 Mysql 组件的 bin 目录运行才能有效。我们编写集成开发环境所用到的命令参数与解释如下：

（1）命令格式：mysqld - install MYSQL - defaults-file=d:\mysql5\my.ini。

含义解释：- install 表示注册服务，MySQL 表示自定义服务名为 MySQL，同样 MySQL 也是自定义的，可以自行修改，如果机器已经默认安装了 mysql 服务，在使用自行开发的系统时，将会照成不必要的冲突，建议自行设的一个。并且这个服务名称，在后续操作中将反复用到。- defaults-file= 表示默认的配置文件，这里需要带入 my.ini 文件的绝对路径。

（2）命令格式：net start MySQL。

含义解释：start 表示启动服务，MySQL 表示自定义服务名为 MySQL。如果之前自行修改过服务名称，这里就是填写自己设定的名称，以后的操作也是相同。

（3）命令格式：net restart MySQL。

含义解释：restart 表示重启服务，MySQL 表示自定义服务名为 MySQL。

（4）命令格式：net stop MySQL。

含义解释：stop 表示停止服务，MySQL 表示自定义服务名为 MySQL。

（5）命令格式：mysqld - remove MySQL。

含义解释：- remove 表示删除服务，MySQL 表示自定义服务名为 MySQL。在执行删除服务操作前，必须先将 MySQL 服务停止后才能移除。

通过以上 5 条常用命令就可以完成 MySQL 的服务注册启动、删除、重启基本操作。

4.2.3 实现组件命令接口功能的技术

通过 PB 的 Run 命令来运行这些组件的接口命令，首先定义一个全局变量 curdir 用来获取主程序所在目录，当然相关的服务组件需要放入这个目录中，才能方便移动。这里通过 apache 组件的注册示例来进行说明，其他命令接口基本与此格

式相同。只需将命令进行修改。

核心代码如下：

```
error_code=Run (curdir+"/bin/httpd-kinstall-n~"APACHE~"",
Minimized! );
If error_code>0then
mle_messages.text =mle_messages.text + "HTTPD 服务~"
APACHE~"注册成功! "+string(now())+"~r~n~r~n";
mle_messages.scroll(mle_messages.linecount());//行滚动
else
mle_messages.text =mle_messages.text + "HTTPD 服务~"
APACHE~"注册失败! "+string(now())+"~r~n~r~n";
mle_messages.scroll(mle_messages.linecount());//行滚动
Endif
```

从以上代码示例可以看出，通过 PB 的 RUN 命令调用 MS-DOS 命令行运行组件的服务，并且值得注意的是 Minimized! 表示该窗口将以最小化的方式运行。当组件命令运行后，可以通过 error_code 命令返回运行状态，当状态为大于 0 时表示运行成功，其他的值表示不成功。

4.2.4 实现动态生成配置文件的技术

PHP 集成环境动态生成配置文件，主要是 Apache、PHP 和 MySQL 文件，在切换 PHP 版本的时候，不需要重新生成 MySQL 和 PHP 的配置文件，所以直接在系统初始化的时候配置文件。

核心代码如下：

```
//初始化 PHP 配置
SetProfileString (curdir + "/PHP/php -5.2/php.ini","PHP","
extension_dir","~"+curdir+"/PHP/php-5.2/ext~"");
SetProfileString (curdir + "/PHP/php -5.3/php.ini","PHP","
extension_dir","~"+curdir+"/PHP/php-5.3/ext~"");
SetProfileString (curdir + "/PHP/php -5.4/php.ini","PHP","
extension_dir","~"+curdir+"/PHP/php-5.4/ext~"");
```

通过 SetProfileString 函数可以直接配置 PHP.ini 的文件，系统初始化时候只需要配置 php 的动态链接库的路径，相关动态链接库开启的配置，可以手动在 PHP.ini 中配置，也可以通过 SetProfileString 函数配置。

核心代码如下：

```
//初始化 MYSQL 配置
SetProfileString (curdir + "/mysql5/my.ini","mysqld","basedir",
curdir+"/mysql5");
SetProfileString (curdir + "/mysql5/my.ini","mysqld","datadir",
curdir+"/mysql5/data");
```

通过 SetProfileString 函数可以直接配置 my.ini 的文件，由于可移动的集成开发环境，使用的 MySQL 官方免安装的数据库版本，所以需要 SetProfileString 函数配置数据库的 MySQL 程序路径和数据路径，其他配置可以保留默认，也可以直接设置。

Apache 配置文件在 PHP 版本切换和实际运用中，将多次使用，所以自定义一个函数 cteate_apache_config () 方便程序，函数中传入两个参数 listenport 和 phpcr，listenport 用于配置 Apache 使用端口，phpcr 用于配置当前选择 PHP 版本。在此只介绍了需要添加或修改核心配置，在实际生成过程中，是需要全部动态生成，直接保留 Apache 配置文件内容就行。还要注意的是之前静态 Apache 配置文件很多地方有 Apache 路径，请使用全局变量 curdir 来动态生成。

核心代码如下：

```
//动态生成 WEB 服务器程序根目录和端口号
String apconfig,tmpconfig;
tmpconfig="ServerRoot~""+curdir+"~"~r~n";
If listenport=0 then
tmpconfig=tmpconfig+"Listen80~r~n";
else
tmpconfig=tmpconfig+"Listen"+string(listenport)+"~r~n";
endif
//根据用户选择配置不同版本的 PHP 配置信息
If phpcr=52 then
/* 生成 PHP5.2 的 apache 配置信息 */
tmpconfig=tmpconfig+"LoadModulephp5_module~""+curdir+"
/PHP/php-5.2/php5apache2_2.dll~"~r~n";//载入 php5.2 的
//apache 动态链接库
tmpconfig=tmpconfig+"PHPIniDir~""+curdir+"/PHP/php-5.2~"
~r~n";//设定 PHP.ini 的目录
tmpconfig =tmpconfig + "LoadFile ~ "" +curdir + "/PHP/php -5.2/
php5ts.dll~"~r~n";
tmpconfig =tmpconfig + "LoadFile ~ "" +curdir + "/PHP/php -5.2/
libmysql.dll~"~r~n";//载入//PHP5.2 目录的 MYSQL 动态链接库
Elseif phpcr=53 then
/* 生成 PHP5.3 的 apache 配置信息 */
tmpconfig=tmpconfig+"LoadModulephp5_module~""+curdir+"
/PHP/php-5.3/php5apache2_2.dll~"~r~n";//载入 php5.3 的
//apache 动态链接库
tmpconfig=tmpconfig+"PHPIniDir~""+curdir+"/PHP/php-5.3~"
~r~n";//设定 PHP.ini 的目录
Elseif phpcr=54 then
tmpconfig=tmpconfig+"LoadModulephp5_module~""+curdir+"
/PHP/php-5.4/php5apache2_2.dll~"~r~n";//载入 php5.4 的
//apache 动态链接库
tmpconfig=tmpconfig+"PHPIniDir~""+curdir+"/PHP/php-5.4~"
~r~n";//设定 PHP.ini 的目录
/* 生成 PHP5.4 的 apache 配置信息 */
else
tmpconfig=tmpconfig+"LoadModulephp5_module~""+curdir+"
/PHP/php-5.2/php5apache2_2.dll~"~r~n";
tmpconfig=tmpconfig+"PHPIniDir~""+curdir+"/PHP/php-5.2~"
~r~n";
tmpconfig =tmpconfig + "LoadFile ~ "" +curdir + "/PHP/php -5.2/
php5ts.dll~"~r~n";
tmpconfig =tmpconfig + "LoadFile ~ "" +curdir + "/PHP/php -5.2/
```

```
libmysql.dll~"~r~n";
/* 生成 PHP5.2 的 apache 配置信息(默认)*/
End if
//设定 WEB 程序运行存储的根目录
tmpconfig=tmpconfig+"DocumentRoot~""+curdir+"/www~"~
r~n";
tmpconfig=tmpconfig+"<Directory~""+curdir+"/www~">~r~
n";
tmpconfig =tmpconfig + "OptionsIndexesFollowSymLinks ~r ~
n";
tmpconfig=tmpconfig+"AllowOverrideNone~r~n";
tmpconfig=tmpconfig+"Orderallow,deny~r~n";
tmpconfig=tmpconfig+"Allowfromall~r~n";
tmpconfig=tmpconfig+"</Directory>~r~n";
//设定默认首页名称和文件类型
tmpconfig=tmpconfig+"<IfModuledir_module>~r~n";
tmpconfig =tmpconfig + "DirectoryIndexindex.phpindex.html ~
r~n";
tmpconfig=tmpconfig+"</IfModule>~r~n";
//增加 PHP 的文件类型的支持
tmpconfig =tmpconfig + "AddTypeapplication/x –httpd –php.
php~r~n";
apconfig=tmpconfig;
Return apconfig;
```

通过以上代码，可以看出动态生成配置文件，就是将动态拼接后的文本返回，并且写入 apache 的 httpd.conf 文件。

4.2.5 实现多版本 PHP 切换功能的技术

PHP 多版本切换主要通过修改 Apache 和切换 PHP 配置文件路径来实现。通过声明一个全局变量来控制 PHP 版本的选择，在动态生成配置文件之前进行判断 PHP 版本，获取选择 PHP 版本的路径，生成相应的配置文件。生成完成配置后，重启 Apache 服务完成，PHP 版本的切换。实现示例如图 2 所示。

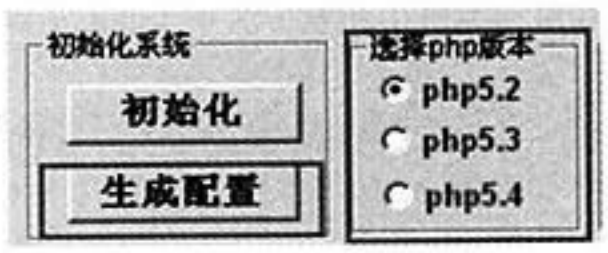


图 2　选择 PHP 版本的界面

声明全局变量：int pcr，在点选按钮的 clicked () 方法中对 pcr 变量进行赋值，达到选择 PHP 版本的目的，如："pcr = 52;"。在生成配置通过 PCR 的值进行判断，并且生成相应的配置文件。

5　结语

系统自投入使用以来，运行稳定，功能符合设计要求，用户操作较为简洁。使得 PHP 测试人员的效率提高，方便了开发人员的工作。该系统仍然需要进一步的完善和用户体验的优化。

图 3　系统初始化界面

参考文献

[1] 由珊珊，栾红娟，孙珍娟. 基于 PB 的设备管理系统设计与开发 [J] . 电脑编程技巧与维护，2012.08:26–28.
[2] 高洛峰. 细说 PHP. 电子工业出版社，2009.
[3] 张长富，李匀. PowerBuilder9.0 参考手册. 北京希望电子出版社. 电子科技大学出版社，2003.

（收稿日期：2012–10–25）

用友 U9 携手签约上海爱普华顿 行业深度应用备受认可

近日，借助用友 U9 在安防、线缆行业的深度应用，用友上海分公司喜签上海爱普华顿电子工业有限公司，充分展现了用友 U9 在此行业中的样板复制能力，继续谱写了 U9 在安防、线缆行业的辉煌！

【企业简介】

爱谱华顿是由华顿国际（香港）投资有限公司与上海爱谱电子线缆系统有限公司共同投资成立的高科技企业。通过多年的努力，爱谱华顿已发展为集生产、研发、销售、服务于一体的综合性高科技企业。建有三个生产基地，市场服务网络遍布国内各主要大中城市。

爱谱华顿产品涵盖弱电线缆、监控器材、综合布线等三大领域，广泛应用于城市建设、房地产业、工矿企业、金融、交通、公安、广播电视、电力电信、文教商贸等行业，以优良的品质和完善的服务为广大用户所肯定。作为国内弱电线缆行业的领先企业，爱谱华顿始终以打造国际化的民族品牌为己任，一贯奉行“持之以恒抓产品质量、实实在在做市场营销、诚诚恳恳为用户服务”的经营理念，始终坚持通过生产规模的不断扩大来实现产品性价比的持续提高。公司将通过技术、服务、管理、文化的持续创新，全力打造行业内的领导品牌。

一种冲击响应方式 UKey 身份认证实现方法

王玉贵

摘　要：介绍一种冲击响应方式 UKey 身份认证实现方法，借以提高信息系统的安全性。

关键词：UKey 应用；身份认证；系统安全；Javascript 脚本

1　引言

网络时代，保障信息系统的安全性成为头等大事。实践证明，在系统身份认证环节引入 UKey (USB key) 应用，通过软硬结合的方式进行身份认证，是一种加强系统安全性行之有效的方法。

在系统身份认证环节引入 UKey (USB key) 应用，最常用的有两种应用方式：一种是基于冲击响应认证方式，另一种是基于 PKI 体系的认证方式。前者和后者比较有不需要获取权威 CA 机构签发的数字证书、不需要建立 CA 服务器、实现方法简单有效等特点，适合在一些规模较小的信息管理系统实施过程中应用。下面将以坚石诚信 ET99 这款产品为例，介绍一种基于冲击响应方式 UKey 身份认证的实现方法。

2　认证过程

2.1　冲击响应方式

所谓冲击响应方式，就是在数据库中和 UKey 硬件中均保存用户密钥信息，用户在系统登录表单中输入 user pin 信息，在浏览器中使用 Javascript 脚本语言进行用户 pin 码验证，相当于登录 UKey 硬件，获得 UKey 读取权利，验证通过后即可读取 UKey 硬件中用户密钥信息，再把从 UKey 中读到的用户密钥信息发送到服务器端，与数据库中保存的用户密钥信息进行比较进而完成用户身份认证过程。

上面所说的认证过程是最基本的基于冲击响应方式的认证过程。UKey 身份认证过程的设计可以灵活多样，本例中使用的认证过程如图 1 所示，具有以下特点：

(1) 系统用户可以分成 UKey 持有用户和非持有用户两种类型，如果不想使用 UKey 认证功能，只需要把用户设置成 UKey 非持有用户类型即可。

(2) 两种用户登录认证操作完全相同，虽然在系统内部这两种用户的认证过程有很大区别，但对用户来说这是后台运行的，用户完全感觉不到。

(3) 使用了用户密码转换用户 pin 码，免除用户输入长 pin 码的麻烦。

2.2　认证过程解释

(1) 用户通过浏览器输入用户名和密码，提交到服务器端进行认证。

(2) 服务器端认证程序从数据库中查询用户信息，返回验证信息包括用户认证结果标志以及 UKey 信息。

(3) 浏览器接收到用户验证信息后，根据其中认证结果标志信息进行判断，如果为 0，认证失败，显示错误信息；如果为 1，表示当前用户为 UKey 非持有用户并且认证成功，进入系统，认证过程结束；如果为 2，表示当前用户为 UKey 持有用户，需要进行下一步操作。

(4) 如果认证结果标志为 2，表示当前用户为 UKey 持有用户，浏览器 Javascript 脚本程序会根据 UKey 信息进行用户 pin 码认证，读取 UKey 中保存的用户密钥信息和 UKey 的硬件序列号 (sn) 信息，调用 UKey 的 MD5HMAC 函数生成客户端加密认证数据，发送到服务器端。

(5) 服务器端从数据库中获取用户密钥信息，调用 HMAC_MD5 算法生成认证加密数据，与客户端发送过来的加密认证数据进行比较，如果一致返回认证成功信息，否则返回认证失败信息。

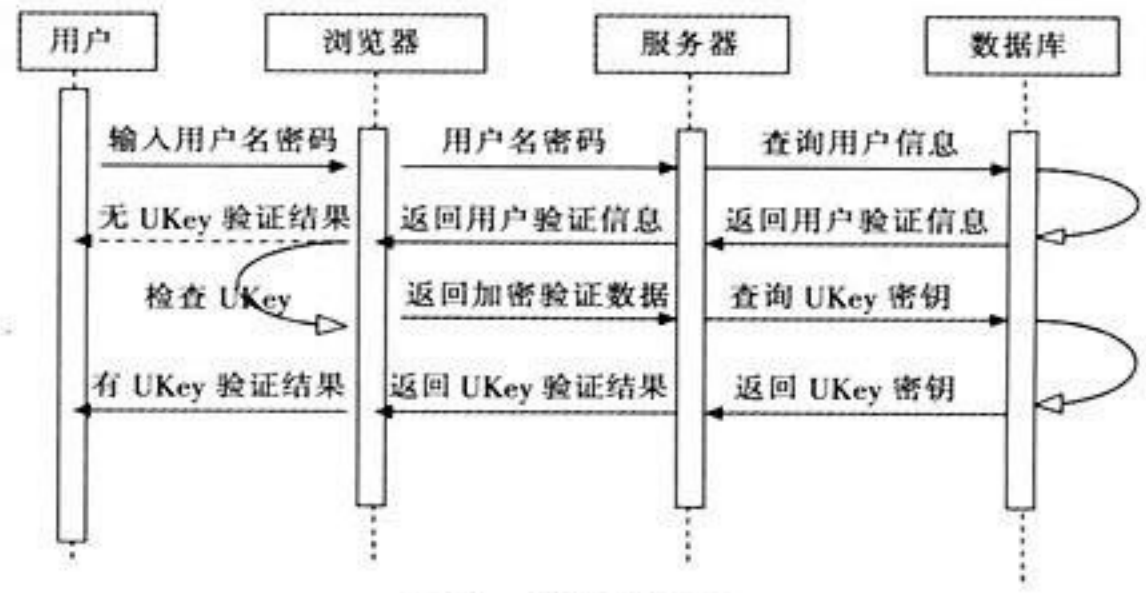


图 1　认证流程图

3　数据库表结构

在此，建立了两个简单的数据表，如表 1 和表 2 所示，用户信息表存储了用户登录用到的用户名和密码信息；UKey 信息表存储了 UKey 信息，loginname 字段设置为外键，与用户信

息表中的 loginname 保持一致。

表 1　User 用户信息表

字段名成	数据类型	说明
Loginname	文本	登录用户名
password	文本	登录密码
Realname	文本	用户姓名

表 2　UKey 信息表

字段名成	数据类型	说明
Loginname	文本	登录用户名
Sn	文本	硬件序列号
Pid	文本	开发商 id
Userpin	文本	用户 pin 码
Seckey	文本	用户密钥

4　实现代码

对认证过程有了初步了解之后，下面介绍实现代码。

4.1　客户端 login.js 源代码

```
var lf=$('#loginform').ajaxForm({beforeSubmit : beforsub,
                dataType:'json',
              success:function(r){
                    switch(r.result){
                       case 0://认证失败
                      alert(r.msg)
                       break;
                      case 1://认证成功
            window.location.href='main.htm';
                          break;
      case 2://持有 ukey,进行下一步操作
                      chkukey(r.ukey);
                    break;
                                        }
                                    }
                    });
function chkukey(k){
          ukey.appendTo('body');
          var uk=ukey[0];
          try{
          uk.FindToken(k.pid);//查找 ukey
          uk.OpenToken(k.pid,1);//打开 ukey
   uk.VerifyPIN(0,k.userpin);//验证用户的 PIN 码,获得权限
          alert("VerifyPIN Success! ");
          var sn=uk.GetSN(),//得到硬件序列号
          hk = uk.Read(0,0,k.hidekey.length);
          if(sn! =k.sn||hk! =k.hidekey){
             alert(' 请检查使用的 UKey 是否是您的 ');
          }else{
             var rd = k.randomData+sn;
var clientDigest = uk.MD5HMAC(1,rd,rd.length);//计算加密认
//证数据
                    ukeyok(clientDigest,k.randomData);
               }
                uk.CloseToken();
             ukey.remove();}
function ukeyok(clientDigest,randomData){
                var u=lf.getVal();
                u.clientDigest=clientDigest;
                u.randomData=randomData;
          //进行第二次信息提交,把 ukey 信息及加密验证数据发
//送到服务器端
                $.ajax({url:'data/system/login',type:'post',
                        dataType:'json',
                        data:u,//包含 ukey 信息及加密验证数据
                        success:function(r){
                            if(r.result==1){
                                window.location.href='main.htm';
                            }else{
                               alert(r.msg);
                            }
                            }});
      }
```

4.2　服务器端 AthenDao.java 代码

```
@Service("authendao")
public class AuthenDao {
      public static final String USER="user";
      private static final String GETUSER="select loginname,
      realname from user where
 loginname=? and password=?";
private static final String GETORG = "select ccode from
user_organ uo,organ o where
 o.id=uo.orgid and uo.loginname=?";
private static final String GETUKEY = "select * from ukey
where loginname=?";
private ConcurrentHashMap<String, HttpSession> sessions =
new ConcurrentHashMap<String, HttpSession>();
      private static final String result="result",msg="msg";
      @Resource(name = "db")
      private SimpleJdbcTemplate db;
public Map<String,Object> authen(HttpSession session,User
u,UKey cukey){
Map<String,Object> r=new HashMap<String,Object>();
            try{
Map <String,Object >  mu =db.queryForMap  (GETUSER,  u.
getLoginname(),u.getPassword());
LoginUser lu=new LoginUser((String)mu.get("loginname"),
(String)mu.get("realname"));
                        String ccode=null;
                       try{
Map <String,Object >  org =db.queryForMap (GETORG,u.
getLoginname());
```

```
                ccode=org.get("ccode").toString();
                lu.setOrgccode(ccode);
            }catch(Exception e1){}
if(cukey.getClientDigest()==null){//第一次提交,没有 ukey 认证信息
                Map<String,Object>ukey=null;
                try{
                    ukey =db.queryForMap (GETUKEY,
                    u.getLoginname());
                }catch(Exception e1){
                }
                if(ukey! =null){ // ukey 持有用户
        String rd=StringUtil.getRandom();
    ukey.put("randomData",rd);
r.put(result,2);r.put(msg, "ukey");r.put("ukey", ukey);
                }else{// ukey 非持有用户
        addUser(session,u,lu);//把用户信息加入 session
            r.put(result,1);r.put(msg, "登录成功");
                }
            }else{//第二次提交,含有 ukey 认证信息
Map <String,Object >ukey =db.queryForMap  (GETUKEY,  u.
getLoginname());
    String rd=cukey.getRandomData()+ukey.get("sn");
                HMAC_MD5 hm = null;
                try {
            //服务器端计算加密认证数据
hm = new HMAC_MD5(ukey.get("seckey").toString().getBytes());
        hm.addData(rd.getBytes());
                hm.sign();
            //与客户端加密认证数据作比较
                if   (cukey.getClientDigest ().equals
                (hm.toString())){
                    lu.setUkey(ukey);
        addUser(session,u,lu); //把用户信息加入 session
        r.put(result,1);r.put(msg, "登录成功");
                }else{
        r.put(result,0);r.put(msg, "ukey 信息错误");
                }
        } catch (NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
        }
        }catch(Exception e){
    r.put(result,0);r.put(msg, "用户名或密码不正确");
        }
        return r;
    }
```

5 Ukey 维护

5.1 Ukey 初始化

Ukey 硬件中存储着开发商 id（pid）、超级用户 pin 码（so pin）、用户 pin 码（user pin）等信息，这些信息都有它的出厂默认设置，并且都是公开的。应用中直接使用这些默认值显然是不安全的，因此使用前需要进行一些初始化设置工作，以保证更高的安全性。

Ukey 初始化设置主要设置开发商 id（pid）、超级用户 pin 码（so pin）、用户 pin 码这几项信息，使用 UKey 厂家提供的工具软件或者通过 UKey 的编程接口（api）都能完成这项工作（如图 2 所示）。需要注意的是开发商 id（pid）是用来识别一个 UKey 是否属于您所在公司的重要信息，在初始化时根据输入的 pid 种子数据生成，使用相同的 pid 种子数据才能生成相同的开发商 id，所以一定要把输入的种子数据保存好，如果种子数据丢失，就不能生成相同的 pid，会造成很大的麻烦。

图 2 通过厂家提供的软件工具设置 UKey 参数

5.2 写入用户密钥

经过初始化的 UKey 需要写入用户密钥信息，为了方便，也为了能够把 UKey 信息保存到数据库中，编程实现了这一操作。

程序在浏览器中用 Javascript 调用 UKey 编程接口，读取 UKey 硬件序列号，写入用户密钥信息，并把这些信息发送到服务器端，保存到 UKey 信息表里。客户端源代码在 initUKey.js 中，服务器端代码在 AuthenDao.java 中，在这里就不详细列出了，如图 3 所示。

图 3 写入用户密钥

经过这一步操作，UKey 就可以发放给用户使用了。

6 结语

之前“用 Spring+jquery 实现一个小系统”一文讲述了如何用使用 Spring+jquery 框架组合实现一个小的信息系统。在此基础讲为系统加入了一种基于冲击响应方式 UKey 身份认证方法，并详细讲述了这一方法的认证过程设计和实现代码，使这个小系统更具安全性和实用性。

（收稿日期：2012-10-24）

在 C# 中执行 DOS 命令探讨

鄢家奇

摘　要：探讨了用 C# 进行 Windows 应用开发时，执行单个或一组 DOS 命令的方法，以及应当关注的问题。同时用一实例，演示了在 C# 中调用 FTP、WinRar 完成较为繁杂的日常事务。

关键词：DOS 命令；批处理程序；WinRar 工具

1　引言

在用 C# 开发 Windows 应用时，有时会需要调用一组 DOS 内外部命令或应用程序，如 DOS 批处理程序、用 FTP 下载文件、用 WinRar 解压 Linux 压缩包（为 *.gz 文件）。

开发的程序需要交给终端用户使用。如果让用户首先执行上述 DOS 命令，再由程序做后续处理，既加大了他们的工作压力，又需要用户掌握这些命令并要保证不漏做、不做错，可行性不高。唯一的办法是由程序将这些命令在运行时自动建立 DOS 批处理程序，然后只需点击命令按纽，一切就 OK 了。这应该是最好的选择。实际上，该方法的运作效果非常令人满意。

2　实现方法

现在，具体探讨如何在 C# 中执行 DOS 命令。

在 C# 中执行 DOS 命令需要使用 System.Diagnostics 名字空间中所定义的 Process 类。当建立了 Process 类的实例后，需要设置实例方法 Start 的相关属性，如当前工作目录、命令行参数等，随后调用该方法就可顺利执行 DOS 命令。

但是，经测试，用此方法（类似单步执行）连续执行（指实例的 Process 进程获取的输人流）的 DOS 命令个数不能超过 3 至 5 个以上，之后的输人流即 DOS 命令就永远处在排队之中而不能被执行。这与启动的 cmd.exe 应用程序的执行环境关系较大，即在 cmd.exe 启动后，在设置属性窗口时配置“命令记录”的“缓冲区大小”、“缓冲区数量”及“丢弃旧的副本”有关。为解决此问题或不受其设置制约，只要将所要执行的 DOS 命令在运行时自动建立成一个临时的批处理程序，那么就不会受到这种制约。

为简化 DOS 命令的执行，设计一个 dos 类，类中仅有一个执行 DOS 命令的静态方法 exeDosCommand，并有 4 种重载形式，可满足多种 DOS 命令执行方式（详所附清单），如单个命令执行、一组命令执行（相当于批处理程序），以及带命令行参数的执行形式。值得注意的是，当进程执行完 DOS 命令后，必须进行适当的延时，以利后续 C# 的处理能够真正地完全感知执行结果。例如，当用此方法执行了 WinRar 解压一个文件后，如果没有延时，那么随即用 Directory.GetFiles 方法就不能即时获取文件清单，给人一种 WinRar 没有执行成功的错觉。

3　实现代码

下面将前文所述用实例做一演示。在 Forms 中添加一命令按纽，其单击事件如下，然后单击按纽即可顺利地执行这组命令。这就比较好地将复杂的、繁杂的工作简单化、自动化了。

```
private void button1_Click(object sender, EventArgs e)
{
string date = "20121021"; // 实际应用时日期可另外获得
    string gzFile = "data_" + date + ".tar.gz", path = "txti";
    // 从 FTP 服务器下载压缩包(为 Linux 文件压缩格式),
//并在后台对其解压
    // 使用 WinRar.exe 而非 Rar.exe 可更好地确保对各种
//压缩文件的解压
     StreamWriter sw = new StreamWriter (path + "
\\gtRmb.ftp", false, Encoding.Default);
    sw.WriteLine("open 192.168.0.1 3322");
    sw.WriteLine("user user pswd");
    sw.WriteLine("bin");
    sw.WriteLine("prompt");
    sw.WriteLine("get " + gzFile);
    sw.WriteLine("bye");
    sw.Close();
    dos.exeDosCommand(new string[] {
    "cd " + path,
    "if exist gtRmb.ftp ftp -n -v -s:gtRmb.ftp > nul",
"if exist " + gzFile + " winrar x -ibck -y -inul -o+ -av- " + gzFile,
    "del gtRmb.ftp",
    "cd .."
    }
  );
}
```

DOS 命令执行类清单，代码如下：

```
using System;
```

```
using System.Diagnostics;
using System.Text;
using System.IO;
using System.Threading;
namespace CuteDos
{
    public class dos
    {
        #region 执行或应用 DOS 外部命令
        /// <summary>
        /// 执行单个 DOS 命令
        /// </summary>
        /// <param name="cmdString">DOS 命令串</param>
        public static void exeDosCommand(string cmdString)
        {
            exeDosCommand(new string[] { cmdString }, "");
        }
        /// <summary>
        /// 执行多个 DOS 命令
        /// </summary>
/// <param name="cmdString">DOS 命令串数组</param>
        public static void exeDosCommand(string[] cmdString)
        {
            exeDosCommand(cmdString, "");
        }
        /// <summary>
        /// 执行单个含命令行参数的 DOS 命令
        /// </summary>
        /// <param name="cmdString">DOS 命令串</param>
        /// <param name="args">命令行参数</param>
        public static void exeDosCommand (string cmdString,
string args)
        {
            exeDosCommand(new string[] { cmdString }, args);
        }
        /// <summary>
        /// 执行多个含命令行参数的 DOS 命令
        /// </summary>
        /// <param name="cmdString">DOS 命令串数组</
param>
        /// <param name="args">命令行参数</param>
        public static void exeDosCommand(string[] cmdString,
string args)
        {
            Process ps = new Process();
            ps.StartInfo.FileName = "cmd.exe";
            ps.StartInfo.UseShellExecute = false;
            ps.StartInfo.RedirectStandardInput = true;
            ps.StartInfo.RedirectStandardOutput = true;
            ps.StartInfo.RedirectStandardError = true;
            ps.StartInfo.CreateNoWindow = true;
            ps.StartInfo.Arguments = args;
            ps.StartInfo.WindowStyle = ProcessWindowStyle.Hidden;
ps.StartInfo.WorkingDirectory = Directory.GetCurrentDirectory
();
            ps.Start();
            if (cmdString.Length >= 3)
            {
    string filename = Directory.GetCurrentDirectory () + "\\" +
Path.GetFileNameWithoutExtension (Path.GetTempFileName
()) + ".bat";
                StreamWriter sw = new StreamWriter(filename,
false, Encoding.Default);
        for (int index = 0; index < cmdString.Length; index++)
                {
                    if (cmdString[index].Length > 0)
                    {
                        sw.WriteLine(cmdString[index]);
                    }
                }
                sw.Close();
                ps.StandardInput.WriteLine(filename);
                DoneDosCommand(ps);
                File.Delete(filename);
            }
            else
            {
    for (int index = 0; index < cmdString.Length; index++)
                {
                    if (cmdString[index].Length > 0)
                    {
     ps.StandardInput.WriteLine(cmdString[index]);
                    }
                }
                DoneDosCommand(ps);
            }
        }
        private static void DoneDosCommand(Process ps)
        {
            ps.StandardInput.WriteLine("exit");
            ps.Close();
            // 延时,等待进程执行完毕
            while (! ps.HasExited)
            {
                Thread.Sleep(1000);
            }
            Thread.Sleep(1000);
        }
        #endregion
    }
}
```

（收稿日期：2012-09-13）

TROUBLESHOOTING OF PROGRAM

Q 如何编程实现为附加在 PE 文件上的口令对话框添加图标

A 笔者曾在本刊 2008 年 6 期上发表过一篇文章：“用 Win32 汇编语言对 PE 格式的 EXE 文件进行口令加密”。在该文中附加在 PE 文件上的口令验证对话框由于没有使用图标资源，因此，被加密的 PE 文件运行时，弹出的口令验证对话框左上角的图标是 Windows 的默认图标（如图 1 所示），这样看起来非常单调，实在是一种遗憾。笔者经过思考并结合自己的编程实践，联想到 Windows 的 PE 文件绝大多数都有自己的图标，仅有极少数 PE 文件没有；因此，口令验证对话框的图标完全可以使用 PE 文件的图标资源来为自己添加图标，不同的 PE 文件图标一般是不同的，这样可使口令验证对话框的图标变成可变的了。

在对话框中使用图标的思路是：第一步，需要在资源文件 RC 中定义该图标标识 ID 和图标文件；第二步，要在对话框的初始化时使用 API 函数 LoadIcon 将图标资源装入（装入图标需要图标的 ID，ID 既可以是数字 ID 名，也可以是字符串名），然后使用 API 函数 SendMessage 向对话框窗口发送设置图标消息，这样对话框左上角就有图标了。因此，要使用 PE 文件的图标资源，最关键的一步就是要获得所需要的图标的标识 ID，经对 PE 文件的资源分析和测试后知道：图标的标识 ID 位于 PE 文件资源类型“图标组”的第二层目录的第一个目录项中；如果图标组有多个图标，也可以用其他图标来作为口令对话框的图标，这里介绍的方法仅使用其第一个图标，因为 PE 文件的图标一般是排在第一个的。

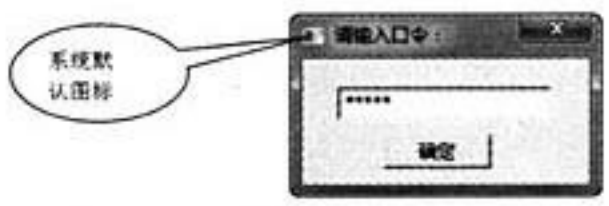


图 1 使用系统默认图标的对话框

(1) PE 文件的资源及组织

PE 文件的资源有光标、图标、菜单、对话框、加速键、字符串等。资源又分为系统预定义资源和自定义资源二个大类，Windows 系统预定义资源共有 23 种类型，除去系统预定义资源外，其余的都属于自定义资源，其中相关的图标资源有 2 种：1）图标（标识为:3,符号名为：RT_ICON）；2）图标组（标识为：14，符号名为：RT_GROUP_ICON）。经过分析测试 PE 文件的资源发现，PE 文件左上角所显示的图标使用的是图标组中的图标，图标和图标组中的图标可以无或一个或多个，最多的可达几百个。主要介绍图标组资源，其他类型资源的组织结构与此类似，当然也可参考其他相关的资料。PE 文件的资源组织与文件目录树类似（但文件目录树并不限制于 5 层），可以分为 5 层，第一层（根目录）为资源类型，如光标、图标、位图、对话框、菜单等等，类型名既可以是 ID 数字名，也可以是字符串名；第二层是资源标识 ID，标识名既可以是 ID 数字名，也可以是字符串名；第三层是资源代码页，第四层是资源数据地址；第五层是资源数据，对图标而言，就是图标数据（如 main.ico 图标文件）。图 2 是图标组资源组织结构（注意：目录与目录项结构体是不同的）。

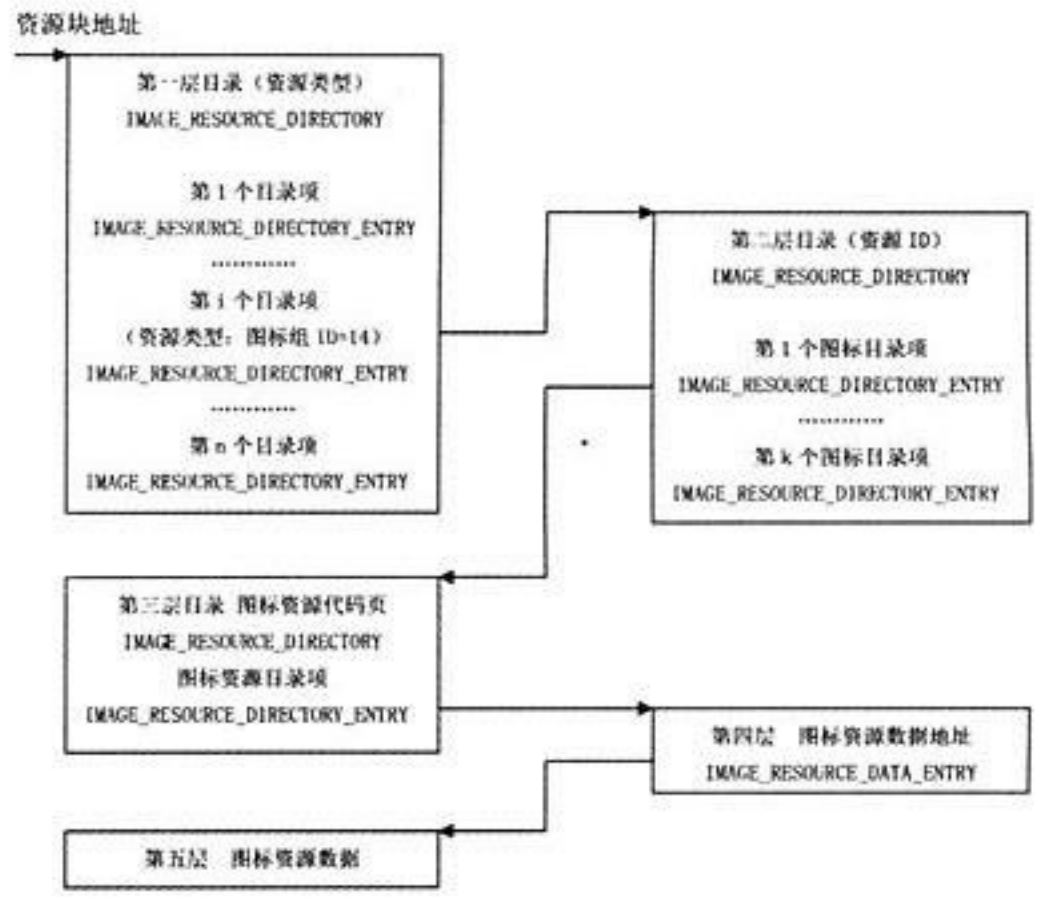


图 2 图标组资源目录树组织结构

从图 2 可以看到第一层目录、第二层目录和第三层目录使用的是同一种结构体，定义如下：

```
IMAGE_RESOURCE_DIRECTORY    STRUCT
Characteristics        dd   ?
TimeDateStamp          dd   ?
MajoyVersion           dw   ?
MinorVersion           dw   ?
NumberOfNamedEntries   dw   ?     ;以字符串名称命名的目录项数量
NumberOfIdEntries      dw   ?     ;以 ID 数字命名的目录项数量
IMAGE_RESOURCE_DIRECTORY    ENDS
```

在这个结构体中，只使用最后二个字段，二个字段的值加起来就是该目录下所有目录项（字符串名+数字 ID 名）的总和n。而第一、二、三层目录中目录项也使用同一种结构体，定义如下：

```
IMAGE_RESOURCE_DIRECTORY_ENTRY    STRUCT
Name1    dd   ?     ; 资源类型字符串名称指针或数字 ID 或代码页编号
OffsetToData    dd   ?     ;地址指针
IMAGE_RESOURCE_DIRECTORY_ENTRY    ENDS
```

对于目录项结构体说明如下：

1） 字段 Name1 在第一层目录中存储的是：资源类型的字符串名称指针（第 31 位值为 1，相对于资源块起始位置的偏移地址）或者是资源类型的数字 ID（第 31 位值为 0），如要获取的资源类型：图标组，它的系统预定义值是数字 ID=14；

2） 字段 Name1 在第二层目录中存储的是：资源标识 ID 的字符串名称指针（第 31 位值为 1，相对于资源块起始位置

的偏移地址）或者是资源类型的数字 ID（第 31 位值为 0）；

3） 字段 Name1 在第三层目录中存储的是：代码页编号；（仅用到资源的第二层目录，后面未用）

4） 当字段 Name1 存储的是字符串名称指针（第 31 位值为 1）时，这时字段 Name1 余下的 31 位组成一个指针，指向一个 IMAGE_RESOURCE_DIR_STRING_U 结构体，定义如下：

```
IMAGE_RESOURCE_DIR_STRING_USTRUCT
Length1 dw   ?     ;Unicode 字符串名称长度
NameString dw   ?     ;Unicode 字符串名称地址指针
IMAGE_RESOURCE_DIR_STRING_UENDS
```

在第二层目录中，从字段 NameString 得到图标资源字符串名地址，可供 API 函数 LoadIconW 装入图标资源。注意：PE 文件的资源字符串名称使用的是 Unicode 编码（简称为宽字符），所以 LoadIcon 函数后面要加上 W（W 表示用于 Unicode 编码）；如需要用 ANSI 编码（简称为 ASCII 码）则要用 API 函数 WideCharToMultiByte 转换一下才能得到，然后可使用 LoadIconA 函数（A 表示用于 ANSI 编码）装入图标资源。LoadIcon 函数的二种形式是不能互换使用的。

5） 在第二层目录中，当字段 Name1 存储的是数字 ID（第 31 位值为 0）时，这就是所要的图标资源的标识 ID，可供 API 函数 LoadIconW 装入图标资源。

6） 字段 OffsetToData 是一个偏移地址（即相对于资源块起始位置的偏移地址），使用时需要加上资源块地址指针，这样得到的地址才是资源项的实际地址。在第一层目录或第二层目录的目录项中，OffsetToData 指针指向下一层的目录（此时第 31 位的值为 1）；在第三层目录的目录项中，OffsetToData 指针（此时第 31 位的值为 0）指向第四层资源数据结构体，定义如下：

```
IMAGE_RESOURCE_DATA_ENTRY  STRUCT
OffsetToData    dd   ?          ;资源数据的 RVA
Size1           dd   ?          ;资源数据的长度
CodePage        dd   ?
Reserved        dd   ?
IMAGE_RESOURCE_DATA_ENTRY  ENDS
```

在这个结构体中，字段 OffsetToData 的 RVA 值+PE 文件载入内存后的基址后，就是指向图标资源数据的绝对地址，更改这里的资源数据，就更改 PE 文件的图标了。另外说一下，微软把 PE 文件的资源的各层目录结构体设计成相同的，把各层目录的目录项结构体也设计成相同的，这样的设计非常适合于用函数递归调用来搜索整个资源树。

(2) 获取 PE 文件图标资源的程序模块

在了解了 PE 文件的图标组的资源组织结构后，下面编程在 PE 文件的资源块中搜索图标资源 。其思路如下：

首先，根据 PE 文件资源块的地址，在第一层目录（资源类型）中，搜索图标组的数字 ID（系统对图标组预定义值为 ID=14，这可以在 Windows.inc 文件中查看），再从图标组所在目录项中取得指向的下一层目录（资源 ID）的地址，从第二层目录中取出第一个目录项的资源 ID 就 OK 了，注意：这个资源 ID 既可以是数字 ID 名，也可以是 Unicode 字符串名的地址。

```
SearchResIcon proc lpResaddress ;lpResaddress——资源块地址
local @dwGroupIcon
pushad
mov @dwGroupIcon,0
mov edi,lpResaddress      ;资源块地址,指向资源的第一层目录
assume edi:ptr IMAGE_RESOURCE_DIRECTORY ;指向目录
;结构体
mov cx,[edi].NumberOfNamedEntries  ;取字符串名命名的目
;录项数量
add cx,[edi].NumberOfIdEntries  ;取数字 ID 命名的目录项数量
add edi,sizeof IMAGE_RESOURCE_DIRECTORY  ; 指向第一个目录项
assume edi:ptr IMAGE_RESOURCE_DIRECTORY_ENTRY
;指向目录项结构体
;在第一层目录的所有目录项中搜索 ID=14 的资源类型:图标组
.repeat
mov eax,[edi].Name1  ;在第一层目录的目录项中获取资源类型
.if ! (eax & 80000000h)      ;是数字 ID 的资源类型
.if eax == 14                ;是图标组资源类型,ID=14
mov eax,[edi].OffsetToData  ;取该目录项所指向的第二层目
;录的偏移地址
and eax,7fffffffh   ;将 31 位的标志位去除,留下后面的是地址
add eax,lpResaddress ;将偏移地址+资源块地址就是第二层目
mov edx,eax                  ;录的实际地址,地址保存在 edx 中
mov @dwGroupIcon,1
.break              ;资源类型:图标组找到后跳出循环
.endif
.endif
add edi,sizeof IMAGE_RESOURCE_DIRECTORY_ENTRY
.untilcxz
.if @dwGroupIcon            ;在第二层目录中获取图标资源 ID
assume edx:ptr IMAGE_RESOURCE_DIRECTORY      ;指向
;第二层目录结构体
add edx,sizeof IMAGE_RESOURCE_DIRECTORY      ;指向
;第一个目录项
assume edx:ptr IMAGE_RESOURCE_DIRECTORY_ENTRY
;指向第一个目录项结构体
mov eax,[edx].Name1  ;取目录项中资源 ID 的名字
.if eax & 80000000h  ;ID 是字符串名称
and eax,7fffffffh    ;去掉第 31 位标志位,留下后面的是地址
add eax,lpResaddress ;偏移地址+资源块地址,指向资源名称
Unicode 字符串结构体
pushad
movzx ecx,word ptr [eax] ;取资源 Unicode 字符串的长度
add eax,2                ;指向 Unicode 字符串的起始地址
mov esi,eax
lea edi,szIconRes  ;取数据段中定义的 Unicode 字符串地址
cld
```

TROUBLESHOOTING OF PROGRAM

```
rep movsw                ;传送 Unicode 字符串
mov dwIconF,1  ;标志 dwIconF=1,表示资源 ID 是字符串名称
popad
.else
mov dwIconID,eax     ;ID 是数字,保存于全局变量中
mov dwIconF,0        ;标志 dwIconF=0,表示资源 ID 是数字
.endif
assume edx:nothing
.endif
assume edi:nothing
popad
ret
SearchResIcon   endp
```

(3) 在口令加密程序中需要加入的代码

在前面获得了 PE 文件的图标资源 ID 后，还需在口令加密程序中加入以下代码后，在被加密的 PE 文件运行后，弹出的口令对话框才能显示出图标。

1) 通过 PE 文件头中的 IMAGE_NT_HEADERS 结构体，取得 PE 文件资源块的 RAV 值。

```
pushad
mov eax,[ebx].OptionalHeader.DataDirectory[16].VirtualAddress
;VirtualAddress 是一个 RAV,指向内存中的资源块
.if ! eax    ;当 PE 文件没有资源时,则跳到后面去执行
jmp @F
.endif
invoke _RVAToOffset[2],@lpMemory,eax     ;将 RAV 转换为磁
;盘文件资源块的偏移地址
add eax,@lpMemory       ;偏移地址+内存映射文件基地址就是
;磁盘文
   ;件资源的绝对地址
invoke SearchResIcon,eax        ;将资源块绝对地址传给搜索
;图标资源程序
popad
@@: …
```

2) 将获得的图标资源 ID（字符串名称地址或数字 ID 和标志 dwIconF）写入被加密的 PE 文件上。

```
mov ecx,@AddCodeFile
add ecx,(offset APPEND_ICON_CODE-offset APPEND_CO
DE)
invoke SetFilePointer,@hFile,ecx,NULL,FILE_BEGIN
mov ecx,158     ;75 个宽字符+数字 ID+标志 dwIconF=158
;个字节
invoke WriteFile,@hFile,offset dwIconF,ecx,addr @dwRet,NULL
```

3) 在口令加密的核心代码中加入以下代码：

```
_ProtoLoadIcon   typedef    proto :dword,:dword
_ProtoSendMessage   typedef  proto  :dword,:dword,:dword,:
dword
;-------------------------------------------------------
_ApiSendMessage         typedef ptr _ProtoSendMessage
_ApiLoadIcon            typedef ptr _ProtoLoadIcon
;-------------------------------------------------------
_SendMessage   _ApiSendMessage  ?
_LoadIcon      _ApiLoadIcon     ?
;-------------------------------------------------------
szLoadIcon  db  'LoadIconW',0  ;注意:使用的是宽字符函数
szSendMessage   db  'SendMessageW',0 ;注意:使用的是宽
;字符函数
;-------------------------------------------------------
APPEND_ICON_CODE  equ  this byte
dwIconFlag  dd  0
appendIcon  dd  0
szappendIcon dw 75 dup (0)     ;Unicode 字符串,有效长度
;74,分析 PE 文件资源发现一般没这么长,要绝对安全可定义
;为 260,但这会增加被加密后 PE 文件的长度。
;-------------------------------------------------------
```

4) 在附加代码的口令对话框的初始化部分的开始处加入下面代码：

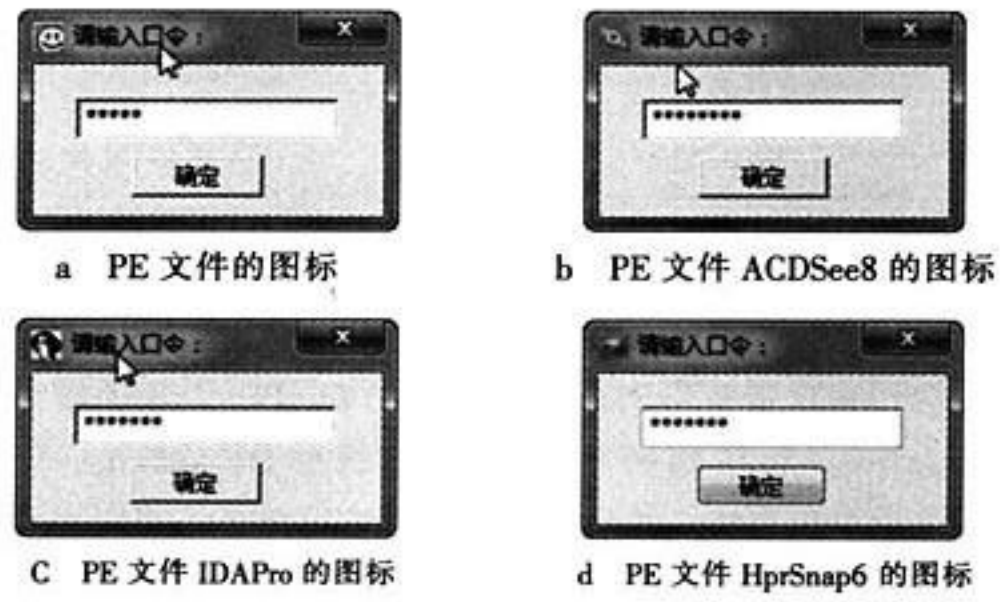
```
mov eax,[ebx+dwIconFlag]
.if  eax              ;是资源数字 ID,还是 Unicode 字符串地址?
lea  eax,[ebx+szappendIcon]  ;是字符串地址
.else
mov  eax,[ebx+appendIcon]      ;是数字 ID
.endif
invoke [ebx+_LoadIcon],[ebx+hInstance1],eax   ;装入图标资源
;向口令对话框窗口发送设置图标消息
invoke [ebx+_SendMessage],hWnd,WM_SETICON,ICON_SM
ALL,eax
;-------------------------------------------------------
```

由于要使用 API 函数 invoke 来调用 LoadIconW 和 SendMessageW，所以需定义对应的函数原形及指针。

5) 动态获取 API 函数 SendMessageW 和 LoadIconW 的代码等，请自行补充。

(4) 添加图标后对话框效果

获取不同 PE 文件图标资源后口令对话框显示的图标，如图 3 所示。



a PE 文件的图标　b PE 文件 ACDSee8 的图标

C PE 文件 IDAPro 的图标　d PE 文件 HprSnap6 的图标

图 3 各种口令加密后 PE 文件运行时，弹出的口令对话框显示图标的实际效果图

(5) 结语

有兴趣的读者在获取 PE 文件图标资源的方法基础上，可用 Win32 汇编或 VC++语言，编程将任一 PE 文件的图标更改为自己的 PE 文件的图标，当然这有一定的难度。曾经流行的熊猫烧香病毒就是这样将被感染的 PE 文件的图标更改为自己的熊猫烧香图标的。

（作者：张钟）

Q 怎样设计与应用软件“忙状态”信息显示面板

A 在软件设计过程中，经常会遇到当软件某个功能进入工作状态后，由于其完成需要较长时间，这时软件进入“忙状态”，给用户的感觉是软件此时无响应，键盘和鼠标等外部设备也停止了工作。对于这种情况，一般的解决办法有两种：方法一：新建一个工作线程，将该功能模块从主线程移到新建的工作线程中；方法二：通过特定界面，为用户提供相应的状态提示，使其明确此时软件正在干什么，从而避免认为软件“死机”。

针对以上情况，给出一个基于 VC 的动态链接库，当软件进入“忙状态”时，开发人员通过调用接口函数，在屏幕中央位置出现一个长条状的显示面板，告诉用户目前软件正在干什么，当“忙状态”结束，再通过接口函数关闭该面板，从而避免了用户的误解。

(1) 设计思想

动态链接库 InfoPanel 采用 VC 实现。只提供一个 stdcall 调用的 API 接口 SetInfoText，原型为：void SetInfoText (const char * lpszText)；参数 lpszText 为要在面板上显示的“忙”信息。调用流程如图 1 所示。

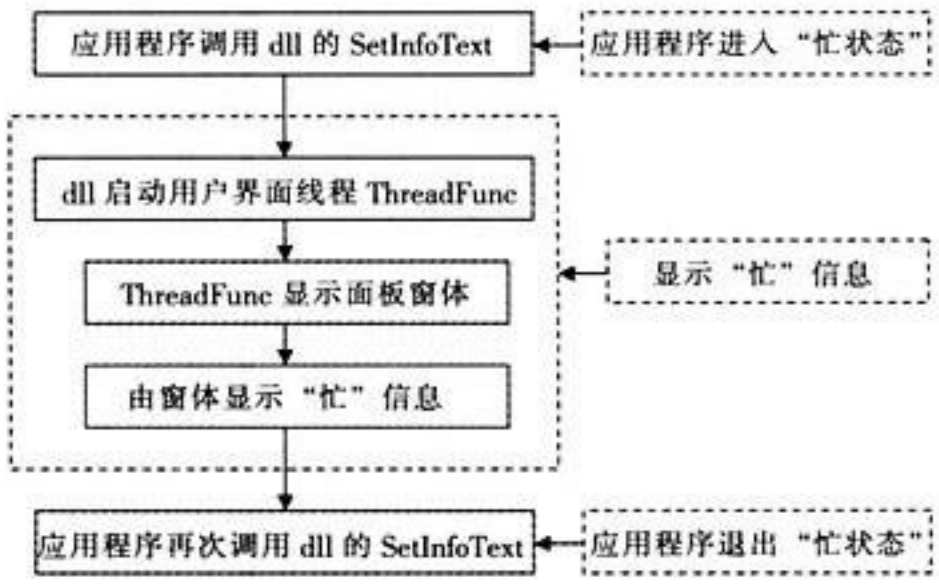


图 1 动态链接库调用流程

当应用程序需要启动面板时，调用 InfoPanel.dll 的输出函数 SetInfoText。这时 dll 将启动用户界面线程 ThreadFunc，线程为用户显示面板窗体，并显示相应的“忙”信息。当应用程序“忙状态”结束，只需再次调用 SetInfoText，并将其中的参数 lpszText 设置为空字符串，这时 dll 将关闭面板窗体，退出用户界面线程，一次完整的调用结束。应用程序可继续后续功能。

(2) 具体实现

```
//全局变量
HWND hWndMain=NULL; //面板窗体句柄
char szInfoText[512]; //"忙"信息文本
HANDLE hThread=NULL; //用户界面线程句柄
HINSTANCE hInst=NULL; //dll 实例句柄
HBRUSH hBrush; //用户面板背景画刷
BOOL APIENTRY DllMain ( HANDLE hModule, DWORD
ul_reason_for_call, LPVOID lpReserved ) //dll 入口函数
{
    hInst=(HINSTANCE)hModule;
hBrush=CreateSolidBrush(GetSysColor(COLOR_INFOBK));
//创建面板窗体背景画刷
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
LRESULT CALLBACK WndProc (HWND hWnd, UINT
message, WPARAM wParam, LPARAM lParam) //面板窗体
//消息处理函数
{
    PAINTSTRUCT ps;
    HDC hdc;
    RECT rc;
    HGDIOBJ hFont;
    switch (message)
    {
    case WM_CREATE:
        return 0;
    case WM_PAINT:
        //对于面板的绘制消息处理
        hdc = BeginPaint(hWnd, &ps);
    GetClientRect(hWnd, &rc);  //获得面板的客户区大小
    FillRect(hdc, &rc, hBrush);  //用背景画刷填充面板背景
        SetBkMode(ps.hdc, TRANSPARENT);
        SetTextColor(ps.hdc, RGB(0,0,0));
        hFont =SelectObject (ps.hdc, GetStockObject
(DEFAULT_GUI_FONT));
DrawText(ps.hdc, szInfoText, -1, &ps.rcPaint, DT_SINGLEL
INE | DT_VCENTER | DT_CENTER); //绘制"忙"状态文本
        SelectObject(ps.hdc, hFont);
        EndPaint(hWnd, &ps);
        break;
    case WM_DESTROY:
        PostQuitMessage(0);
```

```
            break;
        default:
return DefWindowProc(hWnd, message, wParam, lParam);
        }
        return 0;
}
DWORD WINAPI ThreadFunc(LPVOID param)  //用户工作线
//程主函数
{
        //创建面板窗体
        WNDCLASSEX wcex;
        wcex.cbSize = sizeof(WNDCLASSEX);
wcex.style= CS_HREDRAW | CS_VREDRAW;
        wcex.lpfnWndProc = (WNDPROC)WndProc;
        wcex.cbClsExtra = 0;
        wcex.cbWndExtra = 0;
        wcex.hInstance = hInst;
        wcex.hIcon = NULL;
        wcex.hCursor = LoadCursor (NULL, IDC_WAIT); 显示
"等待"光标
        wcex.hbrBackground= hBrush;
        wcex.lpszMenuName       = NULL;
        wcex.lpszClassName= "lrxinfopanel";
        wcex.hIconSm          = NULL;
        RegisterClassEx(&wcex);
        HWND hWnd;
        int x,y,cx,cy;
        cx=600;cy=60;  //面板窗体的大小采用预定义大小,位置
//在屏幕正中央
        x=(GetSystemMetrics(SM_CXSCREEN)-cx)/2;
        y=(GetSystemMetrics(SM_CYSCREEN)-cy)/2;
        hWnd = CreateWindowEx ( WS_EX_NOACTIVATE |
WS_EX_TOPMOST,  "lrxinfopanel",  "",  WS_POPUP  |
WS_DLGFRAME | WS_DISABLED, x, y, cx, cy, NULL,
NULL, hInst, NULL);
        if (! hWnd)  return 0;
        hWndMain=hWnd;
        ShowWindow(hWnd, SW_SHOW);  //创建成功后显示面
//板窗体
        UpdateWindow(hWnd);
        //进入面板窗体消息循环
        MSG msg;
        while (GetMessage(&msg, NULL, 0, 0)! =0)
        {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }
        DeleteObject(hBrush);
        return 0;
}
void SetInfoText(const char * lpszText)  //dll 输出函数
{
        if (lpszText==NULL || strlen(lpszText)==0)  //当调用参数
//为空时,终止线程
        {
                TerminateThread(hThread, 0);
                return;
        }
        if (strlen(lpszText)>sizeof(szInfoText))  //确保有效长度的
//"忙"信息文本
                strncpy(szInfoText, lpszText, sizeof(szInfoText));
        else
                strcpy(szInfoText, lpszText);
        //当线程没有退出时,表明应用程序正在上一个"忙状态"
//中,此时只需更新窗体文本即可
        DWORD dwCode;
        if(GetExitCodeThread(hThread, &dwCode)==TRUE)
        {
                if (dwCode==STILL_ACTIVE)
                {
                        InvalidateRect(hWndMain,NULL, TRUE);
                        return;
                }
        }
        //当所有有效验证通过后,创建用户界面线程
        DWORD dwThreadId;
        hThread  =  CreateThread  (  NULL,0,ThreadFunc,0,0,
&dwThreadId);
}
```

(3) 调用

这里给出在 VB.Net 中的调用示例。

声明如下:

```
    <System.Runtime.InteropServices.DllImportAttribute  ("
InfoPanel.dll")> _
   Public Sub SetInfoText(ByVal szText As String)
    End Sub
```

当程序进入"忙状态"后,显示面板时可如下调用:

```
SetInfoText("正在连接远程数据库……")
```

当“忙”状态结束后,关闭面板时如下调用:

```
Setinfotext("")
```

以上代码在 VS2003 中调试通过。

(4) 结语

设计通过动态链接库将软件“忙状态”信息显示处理逻辑进行了封装,使开发人员可以将主要精力用于软件主体功能开发,从而大幅提高开发效率;对用户而言,也可在软件处于“忙”状态时,从界面信息显示上知道软件在干什么,避免不必要的误解。上述设计在使用后取得了良好的效果。

(作者:刘仁轩)

电脑系统维护经验与技巧

怎样将两个小硬盘合成一个大硬盘

可以使用 Windows 7 磁盘管理工具中的跨区卷功能来合二为一。例如，将两块容量分别为 320GB、200GB 的移动硬盘当作从盘挂接在主机上。操作如下：

右键单击“我的电脑”，并选择“管理”，打开计算机管理控制台。在计算机管理中，单击“磁盘管理”，这时可以看到磁盘 1（298.09GB）和磁盘 3（186.31GB）的状态。在合并前需要将这两个硬盘的所有分区删除。

右键单击磁盘 1，在菜单中选择“New Spannned Volume”新建跨区卷，接着就会出现创建的跨区卷的向导，添加需要合并磁盘 3，并输入想要分配给该卷的空间（建议用默认值），并单击“下一步”。然后根据屏幕提示完成向导，其中包括选择一个驱动器盘符和格式化新卷。

完成向导后，可以看到磁盘 1 和磁盘 3 出现了紫色标记，盘符都为 E 盘 ，这说明它们已经合并成一个新卷了。但是这种操作需要注意的地方是：

将两个小硬盘合并成一个大硬盘使用，对于海量存储带来一定的方便，这样也可以让老硬盘发挥一下余热。但是从卷的结构而言，如果两个硬盘中一个硬盘出现了故障，那么整个卷的数据将被损坏，因此建议用户经常备份数据。

何谓磁盘跨区卷

一个跨区卷是一个包含多块磁盘上的空间的卷（至少两块，最多 32 块磁盘），向跨区卷中存储数据信息的顺序是存满第一块磁盘后再逐渐向后面的磁盘中存储。通过创建跨区卷，可以将多块物理磁盘中的空余空间分配成同一个卷，从而利用了资源。不过跨区卷并不能像 RAID 那样提高性能或容错。

如何擦除硬盘分区数据

如果要彻底删除整个分区的数据，比如用户如果计划将本本卖给别人，或者是本本出现故障要拿去维修，这就要将硬盘分区的重要数据擦除，此时可借助 Ccleaner 软件（下载地址：http：//www.icpcw.com/bzsoft）。运行后切换到“选项”界面，单击“设置”选项，根据需要配置硬盘擦除选项即可。

切换到“工具”界面，单击“驱动器擦除器”选项卡，在“驱动器”下选择需擦除的硬盘分区，建议仅擦除存放重要数据的硬盘分区，以节省擦除时间。而在“擦除”下，如果选择“整个驱动器”，此时会擦除系统分区之外的所有分区。

在“安全”下选择擦除次数，如果只是个人照片或普通文档，建议选择“简单覆写 1 遍”或“高级覆写 3 遍”即可，这样很快就能完成擦除工作。如果是重要的商业文件，例如商业合同、企业报表等，建议选择“复杂覆写 7 遍”或“超复杂覆写 35 遍”，最后单击“擦除”按钮，等待硬盘分区擦除后，所选硬盘分区的数据再也无法恢复了。

如何正确使用西数绿盘

西数绿盘具备 Advanced Format 技术。所谓 Adanced Format 技术，其实是一种将硬盘扇区容量提高为 4KB 的技术。在传统的扇区分割机制中，每 512Byte 的数据之间，就需要间隔一个同步/分隔区域和一个 ECC 错误校验区域。而在 Advanced Format 模式下，每 4KB 才需要一个主样的扇区，相当于把以前的 8 个扇区合而为一，只需要一个同步/分隔区域和一个容量稍大的 ECC 校验区即可。目前来看，包括西部数据的 1TB、1.5 TB、2TB 这 3 大容量硬盘产品，均开始支持高级格式化技术。不过、注意的是，该技术需要 Windows 7 等最新操作（Advanced Format）系统才能够完美地支持。如果仍使用 Windows XP 系统，则需要下载安装 WD Align 软件，才可以正常使用。

怎样高效整理优化本本硬盘

本本用了一段时间以后，由于每天频繁操作，硬盘速度越来越慢，而对磁盘碎片进行整理可改善此问题，但面对本本硬盘容量大，Windows 系统内置的磁盘整理功能不太给力，而借助于 Disk SpeedUP 工具可以高效整理硬盘，并大幅优化一本磁盘性能。具体操作如下：

下载并安装 Disk SpeedUP 工具（下载地址：http：//www.icpcw.com/bzsoft），运行后，进入“Settings”界面，并在“Language”下选择“Chinese”即可显示中文。如想立刻整理硬盘，只需切换到“开始整理”选项卡，单击“开始整理”按钮，选择需要整理的分区盘符，然后单击“开始整理”下拉列表，选择“整理”或“优化”即可。

单击“自动整理”按钮，选择系统所在的分区盘符（如 D 盘），勾选第一个复选框，将时间调至 30 分钟，这样当系统空闲时间超过 30 分钟后，软件则会对所需硬盘分区进行自动整

理。同样勾选第二个复选框，让软件根据 CPU 或硬盘使用率情况自动整理硬盘。

值得一提的是，如果开启自动整理功能，建议进入“设置”选项卡，在“通用”界面勾选“系统启动时自动运行”。切换到“整理”界面后，根据需要配置文件自动整理条件，例如当碎片大小 30MB 或碎片数超过 20 个，软件就会自动整理硬盘。

对于日常办公来说，根据上述方法整理硬盘后，可大幅提升硬盘读写速度。但只是日常家用，建议每隔 1 至 2 月整理一次硬盘即可。这个时候建议开启定时整理功能，方法是单击“计划任务”按钮，勾选“定时运行”复选框后，根据需要设置整理计划即可。

怎样快速整理本本大硬盘磁盘碎片

本本使用很长一段时间后，会发现程序反应速度变慢，这是因为在安装、卸载软件或拷贝数据的过程中，产生了大量的磁盘碎片，Defraggler 工具可以快速高效地整理某个磁盘、文件夹甚至一个文件，效率高，操作简单，非常适合用来整理如今配备大容量硬盘的本本。下载并安装 Defraggler 工具（下载地址：http//www.icpcw.com/bzsoft），在安装过程中，建议取消勾选取“Windows 磁盘整理工具”，以免造成系统错误。运行软件后，若单击磁盘分区盘符，则会显示该分区健康状态，例如 GOOD 表示工作状态良好。

在整理磁盘碎片之前，建议单击“分析”按钮，很快会显示分析结果，例如碎片文件数量、碎片总量等。若要整理磁盘碎片，只需单击“整理”按钮，接着单击“是”按钮即可。据测试，对于 100GB 容量的分区，该软件只需 2 分钟即可整理完毕。

值得一提的是，如果在晚上整理磁盘，由于硬盘容量较大（例如 1TB），且碎片很多，此时可能要花很长时间整理，建议在“设置”菜单下选择“整理完成后关机”，这样用户无需等待，即可让本本自动关机了。

此外，用户也可以设置软件开机自动整理碎片，或是设置一个磁盘整理计划，方法是在“设置”菜单下选择“计划”，选择需要整理的分区，例如系统分区或应用软件安装分区，并设置每天、每周或每月的整理计划，这样就可以让本本无人值守整理磁盘碎片。

怎样处理使用延长线后移动硬盘降速的问题

如果 USB 闪存使用 USB 延长线没有什么问题，而用在移动硬盘上感觉速度明显降低，首先要怀疑延长线的 USB 标准出了问题。首先需要明确的是，要想实现 USB3.0 高速传输，除了存储设备、主板必须支持 USB3.0 规范才行，三者缺一不可。如果用户使用的延长线是 USB2.0 标准的，就会造成降速的问题。建议用户花 10 元左右（1 米延长线的售价）购买一条蓝色的 USB3.0 延长线，问题应该就可以解决了。

如何处理移动硬盘分区丢失的问题

的如果出现插上数据线后，在“我的电脑”中找不到分区盘符，可先将硬盘换到另一个可以正常使用的硬盘盒中进行连接，如果还是出现同样的问题，说明硬盘的主引导扇区可能存在故障。建议先用最新版的 DiskGenius 软件备份分区表（一定要先进行这步操作），然后进行“重建分区表”操作，如果主引导扇区有坏道，得用 victoria 之灯的软件进行“扇区重映射”。如果一切顺利的话，移动硬盘中的数据丝毫不会损坏。否则就得用 EasyRecovery 之类的软件抢救硬盘中的文件。当然，以上方法都是在软件能正确识别硬盘的基础上进行的，操作过程存在一定风险，如果没有什么把握的话，建议找有经验的人帮忙。

怎样处理硬盘只显示系统 C 盘而其余分区无故消失的问题

如果开机发现除了 C 盘能显示并正常打开之外，其余的 D、E、F 等盘都不显见了，这种情况应该是分区表错误导致分区丢失，分区内尚有重要数据，不可“病急乱投医”盲目操作，可以先用一些软件来重建、修复分区表，如《易我分区表医生》、DiskGenius 等，重建成功找回所有分区之后一定记住全盘扫描查杀一下病毒。也里也再次提醒用户，使用电脑一定要养成定期备份重要数据的良好习惯，包括硬盘分区表的备份，防患于未然毕竟比亡羊补牢更好些。

如何处理硬盘坏簇的问题

硬盘出现坏簇后，如果格式化仍然标记有坏簇，那就可能是真正的物理损坏了。可以用分区软件将出现坏道的空间单独分为一个区。在分区前要给坏道留有一定的空间，以免在坏道附近的读写导致坏道扩散。接着再把这个分区隐藏起来，避免再次访问就可以了。此外，还可以下载一个叫“坏盘分区器”（FBDISK）的小软件，用这款软件可以进行坏道自动检测、自动分区、自动隐藏，非常方便。不过，硬盘出现坏道后经常会出现扩散问题，建议尽快购买新硬盘。

硬盘噪音很大怎么办

如果觉得CPU 空闲时硬盘噪声太大的话，可以进入“控制面板”中找到“电源选项”，并设置为“5 分钟之后关闭硬盘”，问题即可迎刃而解了。此外，如果觉得非空闲时噪音也很大的话，则建议登录硬盘厂商的官网，下载降速软件，噪音问题即可解决。但是降速软件对硬盘的性能也会造成影响。

书名：雷神的微软平台安全宝典
ISBN：978-7-302-30743-3
定价：58 元
作者：（美）马伦（Mullen, T.）著

以有效、便捷的方式构建安全防护的基础设施，本书为读者学习微软平台相关的安全技术提供了“一站式”解决方案，这些技术可以用于部署一些典型的基础安全设施。从多个层面详细描述了安全相关的概念和方法论，包括服务器、客户端、组织结构、特定平台相关的安全选项、应用程序相关的安全特性（IIS、SQL、活动目录等），还包括一些新的、从来没有发布过的安全工具，这些安全工具都带有完整的源代码。

书名：WebMatrix ASP.NET Web Pages 开发入门经典
ISBN：978-7-302-30500-2
定价：58.00 元
作者：（英）布林德（Brind, M.），（荷）史潘加斯（Spaanjaars, I.）著

本书采用面向任务的方式，通过实践展示了为什么 WebMatrix 是使用 ASP.NET 开发 Web 网站的理想入门级工具。书中介绍了 WebMatrix 是什么，它的工作原理以及如何发挥它的极大潜力等基础内容，还介绍了成为成功的开发者所需要掌握的其他技术，包括 HTML、CSS 和 SQL。WebMatrix 集成了开源的 Web 应用库以及有价值的代码和数据库支持。更为重要的是，即便你没有 Web 开发经验，现在也可以很自信地使用 WebMatrix 进行开发。

书名：UML 2.2 面向对象分析与设计：第 4 版
ISBN：978-7-302-30424-1
定价：69.80 元
作者：（英）班尼特（Bennett, S.），（英）麦克罗布（McRobb, S.），（英）法默（Farmer, R.）著

本书对前一版进行了修订，反映了信息系统开发中前沿的方法。Bennett、McRobb 和 Farmer 撰写的这本书是面向对象分析与设计领域的经典图书

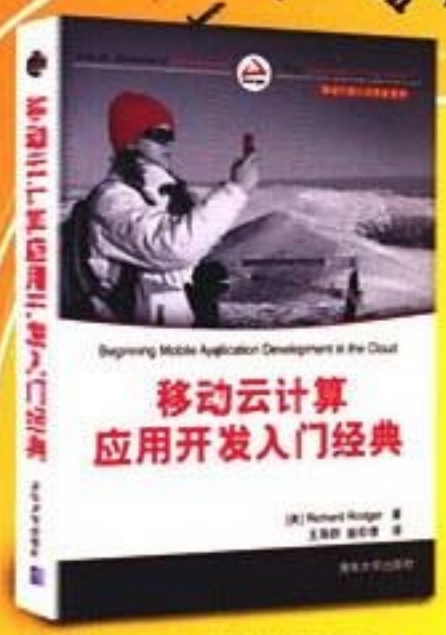
本书为使用 UML 2.2 中的主要技术进行面向对象分析与设计给出了清晰实用的结构，遵循迭代和增量型方法（它们基于业内标准的统一过程），将系统分析和设计置于整个系统生命周期的背景中。本书分为 4 部分：第 1 部分为信息系统的分析和设计以及面向对象提供了背景；第 2 部分关注需求获取和系统分析活动以及 UML 的基本标记法；第 3 部分涉及系统架构和设计活动，以及对象设计的 UML 标记法；第 4 部分介绍系统的实现，如何组织系统生命周期，以及如何开发可重用组件。

书名：JavaScript 宝典（第 7 版）
ISBN：978-7-302-30322-0
定价：128.00 元
作者：（美）Danny Goodman，（美）Michael Morrison，（美）Paul Novitski，（美）Tia Gustaff Rayl 著

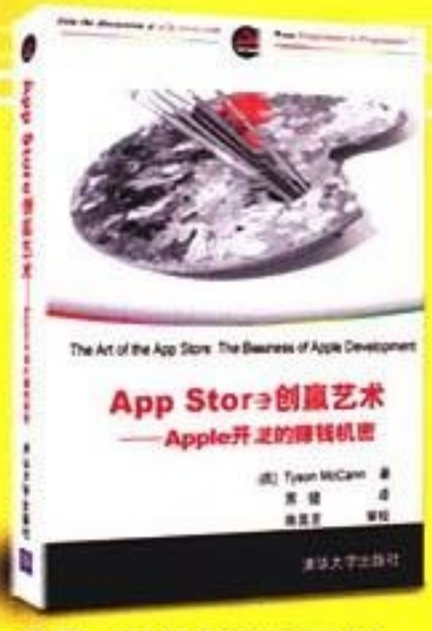
经典畅销书 JavaScript 宝典（第 7 版）在大量示例代码和可运行脚本的引导下，指导您快速掌握 JavaScript 基础知识，并制定出适合预定网站目标的策略。通过认真研读本书，您将可以编写脚本来实现翻转等效果，并熟练使用 Web 2.0 和 JavaScript 子例程库等功能。

本书探讨全新的 JavaScript 编程技术；制定脚本编写策略并选择合适的工具；深入了解终止器、生成器和迭代器；应用最新的 JavaScript 异常处理和自定义对象技术；利用 DOM 的强大功能；使用 Ajax、E4X/XML 和 JSON 推动 Web 2.0 应用；执行数据输入验证和提高安全性

移动开发译丛 全新上市

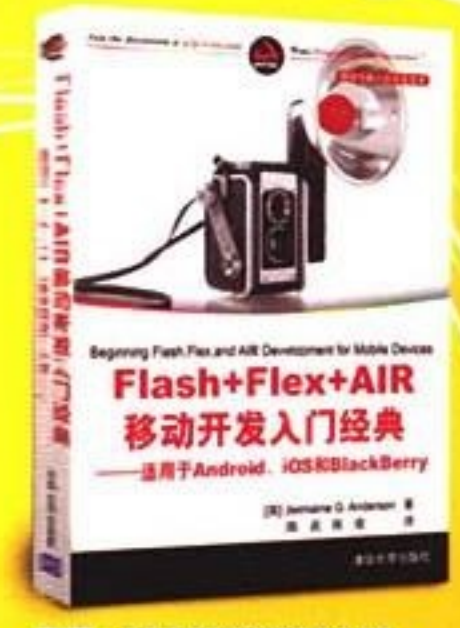


书号: 9787302306580
定价: 59.00元



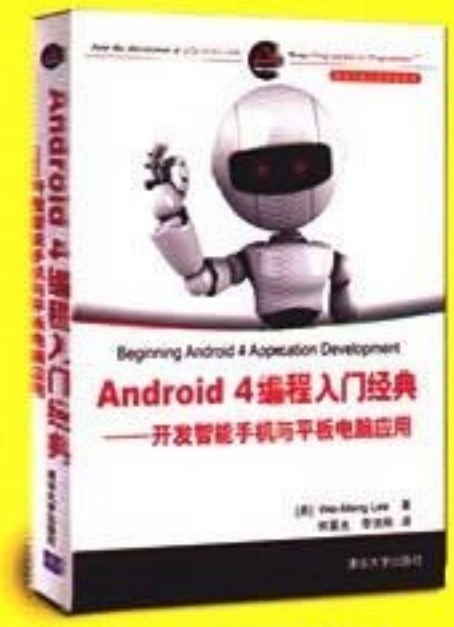
书号: 9787302305019
定价: 59.00元

书号: 9787302306566
定价: 78.00元

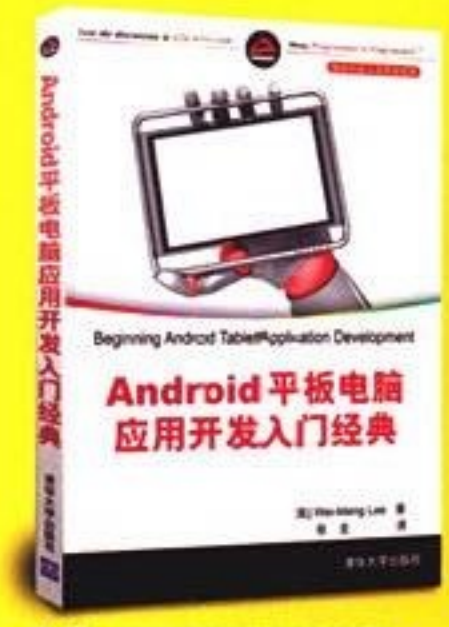


书号: 9787302295440
定价: 49.00元

书号: 9787302299943
定价: 69.80元

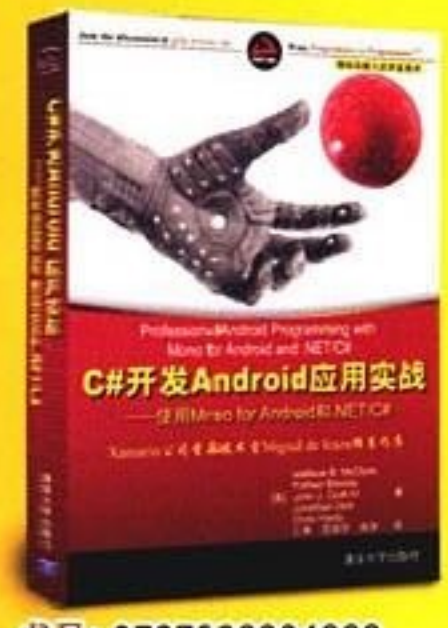


书号: 9787302301516
定价: 68.00元

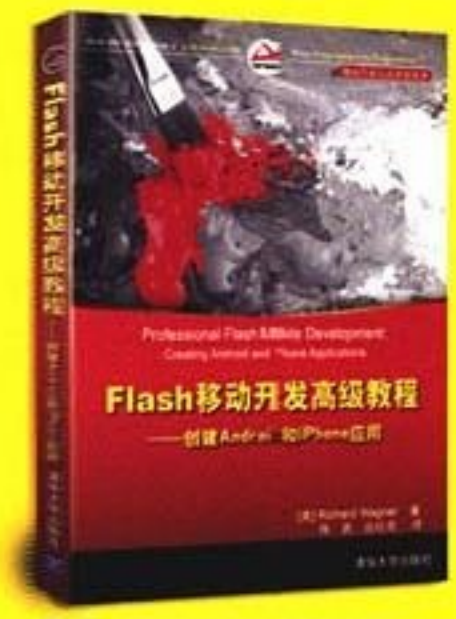


书号: 9787302299967
定价: 49.00元

书号: 9787302303053
定价: 68.00元



书号: 9787302304999
定价: 59.80元



书号: 9787302282099
定价: 48.00元

书号: 9787302304982
定价: 48.00元

书号: 9787302307440
定价: 39.00元

书号: 9787302303039
定价: 59.80元

书号: 9787302301059
定价: 69.00元

书号: 9787302230507
定价: 48.00元

书号: 9787302278894
定价: 48.00元

清华大学出版社
http://www.tup.com.cn

邮发代号：82-715

欢迎订阅 2013 年

《电脑编程技巧与维护》半月刊

上半月刊解析主流编程语言典型编程案例，提供编程实践中高手们的经验与技巧。

下半月刊荟萃电脑产学研应用，展现多领域新进展、新方法、新成果。

上、下半月每期均为 11 元

1. 订阅全年（24 期），可享受 8.5 折优惠，原价 264 元，优惠价 225 元。
2. 单独订阅上、下半月（12 期），可享受 9 折优惠，原价 132 元，优惠价 119 元。

官方网址：http://www.comprg.com.cn

订阅方式

汇款地址：北京市海淀区长春桥路5号6号楼12C9室　收款人：电脑编程技巧与维护杂志社　邮编：100089
电话/传真：82561614　E-mail:zzsfx@vip.sina.com　QQ:565699495
汇款未注明所购买数量和邮寄地址，请与杂志社联系。